SEMI-SUPERVISED LEARNING AND RELEVANCE SEARCH ON NETWORKED DATA

BY

MING JI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair & Director of Research
Professor Dan Roth
Professor Thomas Huang
Associate Professor Yuguo Chen
Associate Professor Jieping Ye, Arizona State University

Abstract

Real-world data entities are often connected by meaningful relationships, forming large-scale networks. With the rapid growth of social networks and online relational data, it is widely recognized that networked data are playing increasingly important roles in people's daily life. Based on whether the nodes and edges have different semantic meanings or not, networks can be roughly categorized into heterogeneous and homogeneous networks. Although homogeneous networks have been studied for decades, some problems still remain unsolved. Heterogeneous networks are much more complicated than homogeneous networks, and have not been explored until recently. Therefore, effective and principled algorithms for mining both homogeneous and heterogeneous networks are in great demand.

In this thesis, two important and closely related problems, semi-supervised learning and relevance search, are studied on both homogeneous and heterogeneous networks. Different from many existing models, algorithms developed in this thesis are theoretically reasonable, widely applicable with minimum constraints, and provide more informative mining results. First, a label selection criterion is proposed to improve the effectiveness of existing semi-supervised learning models on networks. Second, ranking and semi-supervised learning are integrated together to improve the informativeness of the results. Third, a relevance search algorithm that fully considers the geometric structure of the homogeneous networked data is designed. Finally, the relevance search problem between different types of nodes on heterogeneous networks is studied, and the proposed solution is applied on a network constructed from unstructured text data. Research results introduced in this thesis provide advanced principles and the first few steps towards a complete and systematic solution of mining networked data.

To my family for all their love.

Acknowledgments

I would like to thank all the people and agencies who give me tremendous support and help during my PhD study in the past few years.

First and foremost, I would like to express my deepest gratitude and respect to my advisor, Prof. Jiawei Han, for his generous help, support and guidance throughout my PhD study. During my research, Prof. Han always provides insightful and visionary discussions, and encourages me to move forward. When I encounter difficulties in my life, he gives me wise suggestions and fully understands my situation. For me, Prof. Han is my advisor in both research and life. Without his support and understanding, I could not have overcome the most painful year in my life, nor would this thesis have been possible.

Moreover, I would like to thank other doctorate committee members, Prof. Dan Roth, Prof. Thomas Huang, Prof. Yuguo Chen and Prof. Jieping Ye, for their invaluable help during my research and constructive suggestions on this thesis.

I sincerely thank Prof. Xiaofei He and Prof. Deng Cai from Zhejiang University, Dr. Qi He from IBM Almaden Research Center, Prof. Rong Jin from Michigan State University, Dr. Jun Yan from Microsoft Research Asia, Marina Danilevsky, Yizhou Sun, Siyu Gu from Beijing Institute of Technology, Binbin Lin from Arizona State University and Tianbao Yang from GE Global Research for their constructive discussions and great support to my research. The outcome of the collaborations has made the very important pieces of this thesis.

I also owe sincere gratitude to my other collaborators including Bolin Ding, Jing Gao, Zhenhui Li, Cindy Xide Lin, Jialu Liu, Lu Su, Lu-An Tang, Chi Wang, Hongning Wang, Yintao Yu, W. Scott Spangler from IBM Almaden Research Center and Chiyuan Zhang from Massachusetts Institute of Technology. Thanks to all the professors and students in the Database and Information System (DAIS) group for their friendly support and stimulating discussions.

Finally and above all, I am indebted to my parents, Xinli Ding and Wei Ji, and my husband Nan Hua. Thank you for being together with me all the time and through all the difficulties. I love you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, data represented in the form of graphs and networks are playing increasingly important roles in real life. Examples include friendship networks in Facebook[1], co-author networks extracted from bibliographic data, webpages interconnected by hyperlinks on the Web, etc. In these scenarios, data instances are connected by edges representing meaningful relationships. I use graphs and networks interchangeably to represent the same concept in this thesis. Networks and feature vectors are two alternatives to represent the data, and the former is often more natural than the latter in many important applications [24]. Even if the data is traditionally represented in a feature space, it is usually helpful to transform the data into a network, or graph structure (for example, by constructing a nearest neighbor graph) to better exploit the intrinsic characteristics of the data. Therefore, mining networked data is of a great interest to both industry companies and academia.

Most of the existing studies about networks mainly work with homogeneous networks, i.e., networks composed of a single type of nodes, as mentioned above. However, heterogeneous networks composed of multiple types of nodes are more general and prevalent in many real world applications. For example, beyond co-author networks, bibliographic data actually forms a heterogeneous network consisting of multi-typed nodes, such as papers, authors, venues and terms. Other examples include biomedical networks among drugs, diseases, targets and MeSH terms, and E-commerce networks composed of sellers, customers, items and tags.

My past and current research focuses on mining both homogeneous and heterogeneous networks, with a main concentration on semi-supervised learning and relevance search of the nodes. Semi-supervised learning is a classical topic aiming at inferring the labels on all the data given some label information on part of the data, while both labeled and unlabeled data are utilized throughout the learning process. When applied to the graph data, it has wide applications including research

[1] http://www.facebook.com/

community discovery, fraud detection, product recommendation, etc. Semi-supervised learning often performs better than unsupervised learning by employing the valuable human labels, and often needs much fewer labels than supervised learning, therefore attracting substantial attention in the past few years. Although semi-supervised learning has been extensively studied in literature, how to further improve the performance of an existing semi-supervised learning algorithm on networked data is still underexplored. During my PhD study, I have developed a variety of techniques to tackle this problem, generating robust and meaningful prediction results on networks with a very small portion of labeled data. Given an existing semi-supervised learning framework on networked data, there are generally two directions to further improve its performance: (1) minimize its expected error by providing it with the best data configuration, such as an informative subset of data to label, which falls into the topic of active learning [36][31], or a good feature representation of the data, which can be cast as a feature selection problem [32]; and (2) make the results more informative by providing a good summary and understanding of the results [37]. I will discuss my progress in both directions in this thesis.

Relevance search is another important function on networked data. Given a query node in a network, we aim to learn a real-valued relevance function f such that for any two nodes v_i and v_j, $f(v_i) > f(v_j)$ if v_i is more relevant to the query than v_j, and vice-versa. Note that the final goal of relevance search is to rank all the nodes according to their relevance to the query node, where the top ranked nodes are closely relevant to the query and therefore are presented to the user. Therefore, relevance search is one kind of ranking problem. In this thesis, we use "relevance search" and "ranking" to denote two different problems, where the former one computes a *relevance* score for each node with regarding to the query node, and the latter one computes a *ranking* score for each node without specifying the query node. Relevance search and semi-supervised learning are actually closely related in literature if we use "relevant" as the meaning of the labels. For example, semi-supervised learning frameworks have been adapted to solve the relevance search problem [90] by treating the query node as the only labeled node. In this thesis, I will introduce a novel relevance search framework on homogeneous networked data, which follows the idea of a recently proposed semi-supervised learning model [47] and fully considers the special requirement of the relevance search problem. A more challenging problem is how to perform relevance search

over heterogeneous networks. This is also addressed in this thesis over a heterogeneous network constructed in the biomedical domain.

The points below highlight the contributions of this thesis:

- I study how to selectively choose nodes to label such that the expected error of a state-of-the-art semi-supervised learning framework on homogeneous networked data can be minimized [36]. A novel variance minimization criterion is proposed to solve this problem. Compared to existing label selection criterion, our algorithm has the advantage of selecting nodes to label in a batch offline mode with solid theoretical support. By employing our proposed approach, the quality of the labels, as well as the performance of the semi-supervised learner on homogeneous networks, is theoretically improved.

- I explore the problem of integrating ranking and semi-supervised classification to improve the semi-supervised classification performance in heterogeneous networks [37]. By letting ranking and semi-supervised classification iteratively enhance each other, the classification results of the nodes are not only more accurate, but are also more informative since the ranking of nodes within each class provide a good understanding and summary of each class.

- I try to propose a better relevance search algorithm on homogeneous networked data under the assumption that each node in the network has a high-dimensional feature representation, and the nodes are sampled from a submanifold embedded in the ambient Euclidean space [38]. Ideally, beyond smoothness, a desirable relevance function should vary monotonically along the geodesics of the manifold. Therefore, the order of the nodes along the geodesics of the manifold could be well preserved. Moreover, according to the nature of the relevance search problem, no sample in the data set should be more relevant to the query than the query itself. So the relevance function should have the highest value at the query node, and then decrease to other nodes nearby. Our proposed algorithm effectively learns a relevance function that has the highest value at the query node, and varies linearly and therefore monotonically along the geodesics of the data manifold.

- I study the problem of discovering strong relevance between heterogeneous typed biomedical entities using a network-based model. Considering the fact that we do not have a heterogeneous network among biomedical entities available for study, we first construct a heterogeneous

biomedical network from massive biomedical text data. Following the meta path philosophy [74], we select the top-k meta paths that are the most useful for our relevance search task. Equipped with meta path constrained relationship contexts, we design a model to compute the strong relevance between two heterogeneous entities, named EntityRel. Our intuition is, two entities of heterogeneous types are strongly relevant if they are connected by many paths with high weight following the selected meta paths.

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the related work. Chapter 3 $\sim$ 6 present my work in semi-supervised learning and relevance search over networked data. Finally, I conclude this thesis in Chapter 7.

Chapter 2

Related Work

As my thesis focuses on improving semi-supervised learning and relevance search on networked data, it is related to the study of *active learning*, *semi-supervised classification*, *ranking*, *Laplacian-based relevance search on homogeneous data*, and *relevance search on heterogeneous data*. A brief overview of these related methods is discussed in this chapter.

2.1 Active Learning

Most of the existing active learners work with data represented by feature vectors [68]. A seminal paper [31] proposes the first manifold-based active learning algorithm, i.e., GRED, which takes into account both the discriminant and geometrical structure in the data. A nearest neighbor graph is constructed to model the intrinsic manifold structure and incorporated into a least squares loss function as a regularizer. The most informative data points are selected by minimizing the size of the parameter covariance matrix. This principle has been successfully applied to image retrieval [30], video indexing [86], and feature selection [32]. Please see [8] for another active learning approach that exploits the features together with the graph structure. However, in some cases, features of the graph nodes are not always available. Some other methods try to select data based on the graph structure and some labeled nodes. Existing approaches have considered selecting the data that the current classifier is the most uncertain [52], the data with maximum expected information gain [88] or maximum expected entropy reduction [50]. Based on the Gaussian random field model [92], an empirical risk minimization framework [93] is proposed to select examples that minimize the empirical risk estimated by the current classifier. One major limitation of these methods [93, 52, 33, 88, 50, 41] is that they have to obtain the labels of the selected nodes in order to select more data, therefore are not applicable when there is no label information provided during

active learning. When labeling an instance requires time consuming and expensive experiments, these methods are much more costly than running a batch offline mode active learner once and perform labeling in parallel [24].

Recently, there are some efforts devoted to designing label selection criteria that use the graph structure only, without feature representation and label information. Intuitively, one tends to select nodes that lie in high-density (unlabeled) regions [41] or the centers of clusters [52], or have high impact (measured by the graph structure) to unlabeled data [71]. However, these intuitive selection criteria do not have theoretical support on optimizing any classifier.

It is worth mentioning that designing active learners on graphs aiming at minimizing the error of a particular classifier has received substantial interest recently [24, 93]. [24] provides theoretical bounds of the prediction error which are related to label smoothness over the graph, justifying the reasonableness of clustering the nodes and then randomly choose one point from each cluster. Compared with existing methods [24, 93, 8, 41], our proposed algorithm [36] has the advantage of directly minimizing the expected error (instead of the upper bound of the error) in a batch offline mode, through reasonably modeling the probability distribution over the graph. Therefore, we do not require the (potentially expensive) label information of the selected data and tedious retraining of the classifier repeatedly.

2.2 Semi-supervised Classification

Semi-supervised classification is an essential tool in analyzing networked data when the label information is available for some nodes [87, 73]. Collective classification [51, 66, 56] has been proposed to employ both the network structure and the feature representation of nodes in the classification task. Since local features may not be always available, Macskassy et al. [53] develop a relational neighbor classifier to classify network-only data by iteratively assigning a node to the majority class of its neighbors. This idea is similar to the label propagation scheme in graph-based classification [89], where each point iteratively spreads its label information to neighbors so as to ensure both local and global consistency. Based on the same label smoothness assumption, Zhu et al. [92] formulate the problem using a Gaussian random field model defined with respect to the graph. And the framework of manifold regularization [6] is proposed for data-dependent regularization that

exploits the geometry of the probability distribution on the labeled and unlabeled data. However, existing algorithms mainly work on homogeneous networks and graphs, and therefore cannot easily distinguish between the type differences among nodes in a heterogeneous network. Recently, the graph-based classification framework has been extended to work on heterogeneous networked data [39]. In this thesis, we enhance the label propagation framework by providing within-class ranking for nodes in the network, which can improve classification results by providing an informative summary of each class [37].

To enhance the quality of classification, boosting, bagging and ensemble methods have been explored in various studies [28]. In particular, boosting methods such as AdaBoost [18] iteratively learn from their classification mistakes by assigning higher weights to nodes which are misclassified in each previous round, until a stable classification state is reached. Like boosting, our proposed method [37] also adjusts the relative importance of nodes in various rounds of classification. However, [37] uses within-class ranking to measure the importance of each node with regard to each class, in contrast to boosting, which estimates the global importance of each node based on classification mistakes.

2.3 Ranking

As data sets with inherent network structures become increasingly prevalent, ranking networked data has received substantial interest in recent years. Two important representative algorithms are PageRank [60] and HITS [40], both of which propagate information throughout the network to compute the ranking score of each node, using different propagation methods corresponding to different ranking rules. These methods mainly work on homogeneous networks. Recently, PopRank [58] was proposed to rank the popularity of heterogeneous web objects via knowledge propagation throughout the heterogeneous network of web objects. This approach considers that different types of links in a network have different propagation factors, which are trained according to partial ranks given by experts. In contrast, we rank nodes according to their importance within each class, rather than within the global set of all the nodes, and the ranking results in turn facilitate more accurate classification.

The newly proposed NetClus [75] algorithm uses a ranking-clustering mutual enhancement

methodology to cluster nodes in heterogeneous networks, which is closely related to my study. Although this method effectively provides a ranking within each cluster, it has some limitations: (1) it can only work on heterogeneous networks with a star schema; and (2) it requires a prior distribution specified by several labeled *representative* nodes of each cluster, and does not work well with arbitrary labeled nodes, which may not be representative. Thus, if we do not know which nodes are representative in a data set, NetClus cannot be used. However, for heterogeneous networks with arbitrary network schema, our proposed algorithm [37] can make full use of label information available for any nodes to generate accurate classification results and informative rankings.

2.4 Laplacian-based Relevance Search

For relevance search methods under the manifold assumption, the most related work is the Laplacian-based relevance search method [90]. Here we give a brief introduction of [90].

Let $\mathcal{M}$ be a d-dimensional submanifold in the Euclidean space $\mathbb{R}^m$. Given n data points $\{x_1, \ldots, x_n\} \in \mathbb{R}^m$ on $\mathcal{M}$, where x_q is the query $(1 \leq q \leq n)$, we aim to learn a relevance function $f : \mathcal{M} \to \mathbb{R}$, such that $\forall i, j \in \{1, \ldots, n\}$, $f(x_i) > f(x_j)$ if x_i is more relevant to the query x_q than x_j, and vice-versa.

The intuition behind [90] is that if two points x_i and x_j are linked together in a graph/network, then their relevance scores $f(x_i)$ and $f(x_j)$ should be close as well. Let $W \in \mathbb{R}^{n \times n}$ be a symmetric affinity matrix of the network, and $y = [y_1, \ldots, y_n]^T$ be an initial relevance score vector which encodes some prior knowledge about the relevance of each data point to the query. Then [90] minimizes the following objective function:

$$J_{MR}(f) = \sum_{i,j=1}^{n} w_{ij}(\frac{f(x_i)}{\sqrt{D_{ii}}} - \frac{f(x_j)}{\sqrt{D_{jj}}})^2 + \mu \sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{2.1}$$

where w_{ij} denotes the element at the i-th row and j-th column of W, $\mu > 0$ is a regularization parameter and D is a diagonal matrix used for normalization with $D_{ii} = \sum_{j=1}^{n} w_{ij}$. The first term in the above function aims to learn a relevance function that varies smoothly along the data manifold. The second term ensures the finally estimated relevance scores to be close to the initial relevance score assignment y.

Let $f = [f(x_1), \ldots, f(x_n)]^T$. The closed form solution of Eq. (2.1) is the following:

$$f^* = (\mu I + L)^{-1} y \tag{2.2}$$

where I is an identity matrix of size $n \times n$. $L = I - D^{-1/2} W D^{-1/2}$ is the normalized Graph Laplacian [14]. We can also obtain the same solution via an iterative scheme:

$$f^{t+1} = \frac{1}{\mu + 1} S f^t + \frac{\mu}{\mu + 1} y \tag{2.3}$$

where each data point spreads the relevance score to its neighbors iteratively.

The above relevance search framework based on Laplacian regularization has enjoyed long-lasting popularity in the community, with successful extensions in content-based image retrieval [29], relevance search on both undirected and directed graphs [1], ranking on multi-typed inter-related web objects [22], ranking tags over a tag similarity graph [48], etc. However, for the Laplacian-based relevance search framework, we do not exactly know how the relevance function varies, and whether the order of the data points is preserved along the geodesics of the data manifold. [91] points out some drawbacks of [90] and proposes a more robust method by using an iterated graph Laplacian. But [91] still works under the Laplacian-based relevance search framework, addressing the smoothness of the function. Besides, the Laplacian-based relevance search framework requires the final relevance estimation results to be close to the initial relevance score assignment y. In the situation when there is no prior knowledge about the relevance of each data point, people usually assign $y_i = 1$ if x_i is the query and $y_i = 0$ otherwise. This might not be very reasonable, and the finally estimated relevance scores are likely to be biased towards the initial relevance score assignment.

2.5 Relevance Search On Heterogeneous Data

As pointed out in [69] [3], the relationships between entities are the heart of the Semantic Web. Substantial efforts are made to develop techniques for searching complex relationships between entities [3][2] [4]. The relationships are often referred to as Semantic Associations. However, those

Semantic Associations studied in Semantic Web are mainly based on the RDF model, therefore are restricted to simple, existing relationships, such as the "purchase" relationship between customers and items, the "work for" relationship between professors and universities, etc. Different from those existing work, we focus on discovering meaningful relevance relationships that do not explicitly exist in any structured data.

Another family of related work is the recommendation systems, which suggest the items that the users are likely to be interested in [67] [83] [23]. Although recommendation also discovers relevance relationships between two different types of entities (users and items), our problem is fundamentally different from the classical recommendation problem. Specifically, we aim to develop a fully automatic approach that does not use any label information, while recommendation systems usually know some users are interested in certain items.

Given a graph, many methods have been developed for estimating relevance between two nodes, with the Laplacian-based relevance search framework described above being a representative on the homogeneous graphs. For heterogeneous graphs, PathSim [74] gives an interesting meta path based similarity measure between two nodes of the same type. HeteSim [70] and Path Constrained Random Walk [46] estimate the relevance between different types of nodes following the random walk framework. However, the original HeteSim algorithm only uses the binary graph, ignoring the weight on the edges, which is shown to be critically important in our experiments. Path Constrained Random Walk favors the popular entities undesirably and ignores the differences of various contexts inherited from various meta paths. More discussion about these methods can be found in Section 6.5.1

Chapter 3

Label Selection in Semi-supervised Learning on Homogeneous Graphs: A Variance Minimization Criterion

3.1 Overview

Substantial efforts have been devoted to the problem of semi-supervised learning on the nodes in a homogeneous graph. On the other hand, labels can be very expensive to obtain in many real-world applications. Label selection, or active learning methods [15] are then proposed to determine which data examples should be labeled such that the learner could achieve higher prediction accuracy over the unlabeled data as compared to random label selection. Here we use label selection and active learning to denote the same problem. The goal of active learning is to maximize the learner's ability given a fixed budget of labeling effort. While many effective active learners have been developed in literature [68], active learning that takes direct advantage of the graph structure in the data has not been explored until recently [24, 8, 93]. As large-scale data sets with inherent graph structures become increasingly prevalent, reasonable and natural active learning criteria on graphs are in great demand.

In this work, we propose a novel variance minimization perspective to active learning purely on the graph structure, without feature representation and label information. Our study is based on the common assumption that the labels vary smoothly with respect to the graph, which is widely used in the graph-based semi-supervised learning literature [13, 9, 26, 61, 7]. Following one of the most popular graph-based learning frameworks [92], we formulate the smoothness assumption by a Gaussian random field over the graph nodes. Theoretical analysis indicates that the Gaussian field over the unlabeled vertices, conditioned on the labeled data, is a multivariate normal whose mean is the prediction of the harmonic Gaussian field classifier [92]. It is interesting to note that the covariance matrix of the Gaussian field over the unlabeled data is not dependent on the class labels, but only on the graph structure. In this way, we propose to select the data points to label

such that the total variance of the Gaussian field over unlabeled examples, as well as the expected prediction error of the harmonic Gaussian field classifier, is minimized. Efficient computation scheme is then proposed to solve the corresponding optimization problem without introducing any additional parameter.

3.2 Minimizing the Expected Error of Semi-supervised Learning on Graphs

3.2.1 The Problem

We define the active learning problem on graphs as follows. Given a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ associated with a weight matrix W, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of data points (without feature representation) with true labels $\mathbf{y} = (y_1, \ldots, y_n)^T$, $\mathcal{E}$ is the set of edges between any two data points in $\mathcal{V}$, and $W = (w_{ij}) \in \mathbb{R}^{n \times n}$ where w_{ij} denotes the weight on the edge between two data points v_i and v_j. Our goal is to find a subset of points $\mathcal{L} = \{v_{p_1}, \ldots, v_{p_l}\} \subset \mathcal{V}$ where $\{p_i\}_{i=1}^{l} \subset \{1, \ldots, n\}$ are the indices of the points that we should label, such that the classifier learned from the labels on $\mathcal{L}$ could achieve the smallest expected prediction error on the unlabeled data, measured by $\sum_{v_i \in \mathcal{U}} (y_i - y_i^*)^2$, where $\mathcal{U} = \mathcal{V} \setminus \mathcal{L}$ and y_i^* is the predicted label for v_i.

Without loss of generality, here we assume that $\mathcal{G}$ is undirected and connected. We allow continuous labels here, and the labels are assumed to vary smoothly over the graph, i.e., $\sum_{i,j} w_{ij}(y_i - y_j)^2$ is small, which is similar to [24].

3.2.2 The Objective Function

Following [92, 93], the label smoothness assumption could be formulated by a Gaussian random field over the graph:

$$P(\mathbf{y}) = \frac{1}{Z_\beta} \exp(-\beta E(\mathbf{y})) \tag{3.1}$$

where $E(\mathbf{y}) = \frac{1}{2} \sum_{i,j} w_{ij} (y_i - y_j)^2$ is the energy function measuring the smoothness of a label assignment $\mathbf{y} = (y_1, \ldots, y_n)^T$ over the graph, β is an "inverse temperature" parameter, and Z_β is a partition function for the normalization purpose.

Without loss of generality, we can arrange the data points chosen to be labeled to be the first

l instances, i.e., $\mathcal{L} = \{v_1, \ldots, v_l\}$, and the rest $u(= n - l)$ examples $\mathcal{U} = \{v_{l+1}, \ldots, v_{l+u}\}$ are unlabeled. Based on the Gaussian random field model, and the constraint that the predictions on the labeled set are consistent with ground truth, i.e., $\mathbf{y}^*_{\mathcal{L}} = \mathbf{y}_{\mathcal{L}} = (y_1, \ldots, y_l)^T$, a standard method is to predict the labels with the highest probability (or equivalently, minimum energy) [92, 93]. Let $L = D - W$ be the graph Laplacian [14], where D is a diagonal matrix and $D_{ii} = \sum_j w_{ij}$. L can be split into 4 blocks according to the l-th row and column:

$$L = \begin{pmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{pmatrix} \tag{3.2}$$

Then the prediction on the unlabeled nodes given by the harmonic Gaussian field classifier is [92]:

$$\mathbf{y}^*_{\mathcal{U}} = -L_{uu}^{-1} L_{ul} \mathbf{y}_{\mathcal{L}} \tag{3.3}$$

where $\mathbf{y}^*_{\mathcal{U}} = (y^*_{l+1}, \ldots, y^*_{l+u})^T$.

It can be proven that the Gaussian field, conditioned on the labeled data, is a multivariate normal: $\mathbf{y}_{\mathcal{U}} \sim \mathcal{N}(\mathbf{y}^*_{\mathcal{U}}, L_{uu}^{-1})$ [93], where $\mathbf{y}_{\mathcal{U}} = (y_{l+1}, \ldots, y_{l+u})^T$. Then we compute the expected prediction error on the unlabeled nodes as follows:

$$\begin{aligned} & \mathrm{E}\left(\sum_{v_i \in \mathcal{U}} (y_i - y_i^*)^2\right) \\ = \; & \mathrm{E}\left((\mathbf{y}_{\mathcal{U}} - \mathbf{y}^*_{\mathcal{U}})^T (\mathbf{y}_{\mathcal{U}} - \mathbf{y}^*_{\mathcal{U}})\right) \\ = \; & \mathrm{E}\left(\mathrm{Tr}\left((\mathbf{y}_{\mathcal{U}} - \mathbf{y}^*_{\mathcal{U}})(\mathbf{y}_{\mathcal{U}} - \mathbf{y}^*_{\mathcal{U}})^T\right)\right) \\ = \; & \mathrm{Tr}\left(\mathrm{E}\left((\mathbf{y}_{\mathcal{U}} - \mathbf{y}^*_{\mathcal{U}})(\mathbf{y}_{\mathcal{U}} - \mathbf{y}^*_{\mathcal{U}})^T\right)\right) \\ = \; & \mathrm{Tr}\left(\mathrm{var}(\mathbf{y}_{\mathcal{U}})\right) = \mathrm{Tr}(L_{uu}^{-1}) \end{aligned} \tag{3.4}$$

In order to minimize the expected error of the prediction results, we should minimize the variance of the statistical learning model [15]. Therefore, we propose to select the nodes to label by solving the following optimization problem:

$$\arg\min_{\mathcal{L} \subset \mathcal{V}} \mathrm{Tr}(L_{uu}^{-1}) \tag{3.5}$$

It is easy to verify that Eq. (3.5) is independent of the order of the examples, but only dependent on the choice of the set of the nodes that we choose *not* to label. Therefore, our objective function is well defined.

3.3 Efficient Optimization

Let $\{q_1, \ldots, q_u\}$ be the indices of the nodes that we choose *not* to label. Following the above discussion, our objective is to select a $u \times u$ submatrix L_{uu} of L on the intersections of the $\{q_1, \ldots, q_u\}$-th rows and columns, such that the trace of L_{uu}^{-1} is minimized. This optimization problem in Eq. (3.5) is challenging since the number of candidate sets for $\mathcal{L}$ is exponential in the total number of examples n. Moreover, since the number of unlabeled examples is usually huge, L_{uu} will likely be a large matrix and directly optimizing Eq. (3.5) based on the set of unlabeled data is very computationally expensive. In this section, we first transform the objective function so that it can be represented by the instances that we choose to *label*, and then propose an efficient sequential optimization scheme.

3.3.1 Formulations

We first construct a selection matrix $S \in \mathbb{R}^{u \times n}$ to help selecting L_{uu} from L as follows:

$$S_{ij} = \begin{cases} 1 & \text{if } j = q_i \\ 0 & \text{otherwise.} \end{cases}$$

Then we have:

$$L_{uu} = SLS^T \tag{3.6}$$

Since L is symmetric, it has the eigendecomposition result as follows:

$$L = X\Sigma X^T \tag{3.7}$$

such that X is an orthonormal matrix, and $\Sigma = \text{diag}\{\lambda_1, \ldots, \lambda_n\}$, where $\{\lambda_i\}_{i=1}^n$ are the eigenvalues of L, and $\lambda_1 \geq \ldots \geq \lambda_n = 0$. Then

$$L_{uu} = SLS^T = SX\Sigma X^T S^T \tag{3.8}$$

Suppose $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, where $\mathbf{x}_i^T$ is the i-th row of X. Let $Q = SX$, then $L_{uu} = Q\Sigma Q^T$. Since S is the selection matrix, then $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_u)^T \in \mathbb{R}^{u \times n}$ consists of the $\{q_1, \ldots, q_u\}$-th rows of X. We further define two sets of vectors $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_u\}$, then our objective function in Eq. (3.5) is equivalent to the following:

$$\arg\min_{\mathcal{Q} \subset \mathcal{X}} \text{Tr}\left((Q\Sigma Q^T)^{-1}\right) \tag{3.9}$$

Let I_n denote the identity matrix of size $n \times n$. By using the Woodbury formula [21], we have the following:

$$\begin{aligned}
& (Q\Sigma Q^T)^{-1} \\
= \; & \left(Q(\Sigma + I_n)Q^T - QQ^T\right)^{-1} \\
= \; & \left(Q(\Sigma + I_n)Q^T - I_u\right)^{-1} \\
= \; & (-I_u)^{-1} - Q\left((\Sigma + I_n)^{-1} + Q^T(-I_u)^{-1}Q\right)^{-1} Q^T \\
= \; & -I_u - Q\left(M^{-1} - Q^TQ\right)^{-1} Q^T
\end{aligned}$$

where $M = \Sigma + I_n = \text{diag}\{\lambda_1 + 1, \ldots, \lambda_n + 1\}$. According to the matrix determinant lemma [27], we have:

$$\begin{aligned}
& \det\left(M^{-1} - Q^T Q\right) \\
= \; & (-1)^n \det\left(-M^{-1} + Q^T Q\right) \\
= \; & (-1)^n \det\left(-M^{-1}\right) \det\left(I_u + Q\left(-M^{-1}\right)^{-1} Q^T\right) \\
= \; & (-1)^{2n} \det\left(M^{-1}\right) \det\left(I_u - QMQ^T\right) \\
= \; & \det\left(I_u - Q\Sigma Q^T - QI_nQ^T\right) \prod_{i=1}^{n} \frac{1}{\lambda_i + 1} \\
= \; & \det\left(I_u - L_{uu} - I_u\right) \prod_{i=1}^{n} \frac{1}{\lambda_i + 1} \\
= \; & \det(-L_{uu}) \prod_{i=1}^{n} \frac{1}{\lambda_i + 1}
\end{aligned} \tag{3.10}$$

As long as $0 < u < n$ and the graph is connected, it can be easily proven that L_{uu} is invertible, and so is $M^{-1} - Q^T Q$. Recall that $\text{Tr}(AB) = \text{Tr}(BA)$, we further have:

$$\begin{aligned}
& \text{Tr}\left((Q\Sigma Q^T)^{-1}\right) \\
= \; & -u - \text{Tr}\left(Q\left(M^{-1} - Q^T Q\right)^{-1} Q^T\right) \\
= \; & -u - \text{Tr}\left(\left(M^{-1} - Q^T Q\right)^{-1} Q^T Q\right) \\
= \; & -u + \text{Tr}\left(\left(M^{-1} - Q^T Q\right)^{-1} \left(-Q^T Q + M^{-1} - M^{-1}\right)\right) \\
= \; & -u + \text{Tr}\left(I_n - \left(M^{-1} - Q^T Q\right)^{-1} M^{-1}\right) \\
= \; & n - u - \text{Tr}\left(\left(M^{-1} - Q^T Q\right)^{-1} M^{-1}\right) \\
= \; & l - \text{Tr}\left(\left(M^{-1} - \sum_{i=1}^{u} \mathbf{q}_i \mathbf{q}_i^T\right)^{-1} M^{-1}\right)
\end{aligned}$$

Let $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_l\} = \mathcal{X} \setminus \mathcal{Q}$ be the $\{p_1, \ldots, p_l\}$-th row vectors of X that correspond to the examples that we choose to *label*, then we have:

$$\begin{aligned}
& \operatorname{Tr}\left((Q\Sigma Q^T)^{-1}\right) \\
= \; & l - \operatorname{Tr}\left(\left(M^{-1} - \sum_{i=1}^{u} \mathbf{q}_i \mathbf{q}_i^T\right)^{-1} M^{-1}\right) \\
= \; & l - \operatorname{Tr}\left(\left(M^{-1} - \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T + \sum_{i=1}^{l} \mathbf{p}_i \mathbf{p}_i^T\right)^{-1} M^{-1}\right) \\
= \; & l - \operatorname{Tr}\left(\left(M^{-1} - X^T X + \sum_{i=1}^{l} \mathbf{p}_i \mathbf{p}_i^T\right)^{-1} M^{-1}\right) \\
= \; & l - \operatorname{Tr}\left(\left(M^{-1} - I_n + \sum_{i=1}^{l} \mathbf{p}_i \mathbf{p}_i^T\right)^{-1} M^{-1}\right)
\end{aligned}$$

Let $A_0 = M^{-1} - I_n$. Since the number of data points to be labeled, l, is fixed, our objective function in Eq. (3.9) reduces to the following:

$$\arg\max_{\mathcal{P} \subset \mathcal{X}} \operatorname{Tr}\left(\left(A_0 + \sum_{i=1}^{l} \mathbf{p}_i \mathbf{p}_i^T\right)^{-1} M^{-1}\right) \tag{3.11}$$

In the following, we describe an efficient sequential optimization scheme to select which nodes we should label in a graph.

3.3.2 Selecting the First Point

Setting $l = 1$ in Eq. (3.11), we obtain the objective function of selecting one (or the first) data point to label:

$$\arg\max_{\mathbf{p} \in \mathcal{X}} \operatorname{Tr}\left(\left(A_0 + \mathbf{p}\mathbf{p}^T\right)^{-1} M^{-1}\right) \tag{3.12}$$

Usually, matrix inversion formulae in the form of $\left(A_0 + \mathbf{p}\mathbf{p}^T\right)^{-1}$ can be simplified using the Sherman-Morrison formula [21]:

$$(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}} \tag{3.13}$$

However, note that

$$\begin{aligned} A_0 &= M^{-1} - I_n \\ &= \text{diag}\left\{\frac{1}{\lambda_1 + 1} - 1, \ldots, \frac{1}{\lambda_n + 1} - 1\right\} \\ &= \text{diag}\left\{\frac{-\lambda_1}{\lambda_1 + 1}, \ldots, \frac{-\lambda_n}{\lambda_n + 1}\right\} \end{aligned} \tag{3.14}$$

is singular since the smallest eigenvalue of L (denoted as λ_n) is equal to 0. Therefore, the Sherman-Morrison formula (3.13) cannot be applied here. In this subsection, we derive how to select the first point to label by performing some modification of Eq. (3.12).

For a connected graph, it is known that all the eigenvalues of L, except λ_n, are larger than 0. The eigenvector corresponding to λ_n is a $n \times 1$ constant vector which can be denoted as $(c, \ldots, c)^T$. So any $\mathbf{p} \in \mathcal{X}$ can be represented as $\mathbf{p} = (\mathbf{v}^T, c)^T$ where $\mathbf{v}$ is a $(n-1) \times 1$ vector after removing the last element of $\mathbf{p}$. Let $B = \text{diag}\left\{\frac{-\lambda_1}{\lambda_1+1}, \ldots, \frac{-\lambda_{n-1}}{\lambda_{n-1}+1}\right\} \in \mathbb{R}^{(n-1)\times(n-1)}$ be the matrix after removing the last row and column of A_0, which is invertible. Hence:

$$\begin{aligned} A_0 + \mathbf{p}\mathbf{p}^T &= \begin{pmatrix} B & c\mathbf{v} \\ c\mathbf{v}^T & c^2 \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}^T & 0 \end{pmatrix} \\ &= \hat{B} + \hat{\mathbf{v}}\hat{\mathbf{v}}^T \end{aligned}$$

where $\hat{B} = \begin{pmatrix} B & c\mathbf{v} \\ c\mathbf{v}^T & c^2 \end{pmatrix}$ and $\hat{\mathbf{v}} = \begin{pmatrix} \mathbf{v}^T & 0 \end{pmatrix}^T$. By doing blockwise matrix inversion, we have:

$$\hat{B}^{-1} = \begin{pmatrix} B^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{c^2(1 - \mathbf{v}^T B^{-1}\mathbf{v})} \begin{pmatrix} c^2 B^{-1}\mathbf{v}\mathbf{v}^T B^{-1} & -cB^{-1}\mathbf{v} \\ -c\mathbf{v}^T B^{-1} & 1 \end{pmatrix}$$

where $B^{-1} = \text{diag}\left\{-\frac{\lambda_1+1}{\lambda_1}, \ldots, -\frac{\lambda_{n-1}+1}{\lambda_{n-1}}\right\}$. Now we can employ Eq. (3.13) and have:

$$\begin{aligned} & \left(A_0 + \mathbf{p}\mathbf{p}^T\right)^{-1} \\ = & \left(\hat{B} + \hat{\mathbf{v}}\hat{\mathbf{v}}^T\right)^{-1} \\ = & \hat{B}^{-1} - \frac{\hat{B}^{-1}\hat{\mathbf{v}}\hat{\mathbf{v}}^T\hat{B}^{-1}}{1 + \hat{\mathbf{v}}^T\hat{B}^{-1}\hat{\mathbf{v}}} \end{aligned} \tag{3.15}$$

Recall that $M^{-1} = \text{diag}\left\{\frac{1}{\lambda_1+1}, \ldots, \frac{1}{\lambda_n+1}\right\}$. Therefore, $\left(A_0 + \mathbf{p}\mathbf{p}^T\right)^{-1} M^{-1}$ can be computed efficiently without matrix inversion for any given $\mathbf{p}$. We select the first data point to label that corresponds to $\mathbf{p} \in \mathcal{X}$ such that Eq. (3.12) is maximized.

3.3.3 Selecting More Points

We define:

$$A_l = A_0 + \sum_{i=1}^{l} \mathbf{p}_i \mathbf{p}_i^T \tag{3.16}$$

Suppose $l(\geq 1)$ data points have been selected, which correspond to the rows of X: $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\} = \mathcal{P}_l \subset \mathcal{X}$, then the $(l+1)$-th instance can be selected by solving the following:

$$\mathbf{p}_{l+1} = \arg\max_{\mathbf{p} \in \mathcal{X} \setminus \mathcal{P}_l} \text{Tr}\left(\left(A_l + \mathbf{p}\mathbf{p}^T\right)^{-1} M^{-1}\right) \tag{3.17}$$

By using the Sherman-Morrison formula (3.13), we have:

$$\left(A_l + \mathbf{p}\mathbf{p}^T\right)^{-1} = A_l^{-1} - \frac{A_l^{-1}\mathbf{p}\mathbf{p}^T A_l^{-1}}{1 + \mathbf{p}^T A_l^{-1}\mathbf{p}} \tag{3.18}$$

And A_1^{-1} can be computed using Eq. (3.15). Therefore:

$$\begin{aligned} & \text{Tr}\left(\left(A_l + \mathbf{p}\mathbf{p}^T\right)^{-1} M^{-1}\right) \\ = & \text{Tr}(A_l^{-1}M^{-1}) - \frac{\text{Tr}\left(A_l^{-1}\mathbf{p}\mathbf{p}^T A_l^{-1} M^{-1}\right)}{1 + \mathbf{p}^T A_l^{-1}\mathbf{p}} \\ = & \text{Tr}(A_l^{-1}M^{-1}) - \frac{\text{Tr}\left(\mathbf{p}^T A_l^{-1} M^{-1} A_l^{-1}\mathbf{p}\right)}{1 + \mathbf{p}^T A_l^{-1}\mathbf{p}} \\ = & \text{Tr}(A_l^{-1}M^{-1}) - \frac{\mathbf{p}^T A_l^{-1} M^{-1} A_l^{-1}\mathbf{p}}{1 + \mathbf{p}^T A_l^{-1}\mathbf{p}} \end{aligned} \tag{3.19}$$

Since $\mathrm{Tr}(A_l^{-1}M^{-1})$ is a constant when selecting the $(l+1)$-th data point, we choose the $(l+1)$-th point to label that corresponds to the following $\mathbf{p}_{l+1}$:

$$\mathbf{p}_{l+1} = \arg\min_{\mathbf{p}\in\mathcal{X}\setminus\mathcal{P}_l} \frac{\mathbf{p}^T A_l^{-1} M^{-1} A_l^{-1} \mathbf{p}}{1 + \mathbf{p}^T A_l^{-1} \mathbf{p}} \tag{3.20}$$

Once $\mathbf{p}_{l+1}$ is obtained, A_{l+1} can be updated according to Eq. (3.18).

3.4 Experimental Results

In this section, we apply our proposed active learning method based on Variance Minimization (denoted as **VM**) in the Gaussian random field to several real-world data sets to test its effectiveness. We use the labels of vertices chosen by different active learning criteria to train a harmonic Gaussian field classifier [92] to predict the labels of the rest of the nodes in the graph. The following five label selection methods are compared:

- Our proposed VM algorithm (**VM**).
- Empirical Risk Minimization (**ERM**) [10].
- Random selection (**Random**).
- Label Selection based on Clustering (**LSC**) [24].
- Uncertainty sampling (**Uncertainty**).

When our budget is to select l instances to label, the **LSC** method clusters the data into l clusters and then randomly select one example from each cluster. This method minimizes the prediction error bound related to label smoothness, and empirically performs the best in [24]. We use Spectral Clustering [57] to cluster the graph nodes. The results of **Random** and **LSC** are both averaged over 10 random trials. **ERM** and **Uncertainty** are two methods that iteratively query more data to label according to the classifier trained by the previously labeled data. **ERM** selects examples that minimize the empirical risk estimated by the current classifier. The **Uncertainty** criterion selects the instances whose labels the current classifier is the most uncertain. Recall that the harmonic Gaussian field classifier adopts the one-against-all scheme in multi-class classification.

Suppose we have k classes and u unlabeled data points, then the classifier outputs a $u \times k$ score matrix, where each row is for an unlabeled point, and each column for a class. The class with the largest value in the i-th row is the predicted class of the i-th unlabeled point. Let $f_1(v_i)$ denote the largest score of node v_i related to a certain class k_1, and $f_2(v_i)$ denote the second largest score of v_i related to a different class k_2. The smaller $f_1(v_i) - f_2(v_i)$, the more uncertain the classifier is about the label prediction of v_i. Therefore, we select new instances $\{v_i\}$ to label with the smallest values of $f_1(v_i) - f_2(v_i)$. This strategy is also compared in [52]. Notice that **ERM** and **Uncertainty** use the label information of the previously selected data, while other active learning methods do not. In order to test them in our scenario that very little (if not none) label information is available during active learning, for **ERM** and **Uncertainty**, we randomly choose an initial set of labels for each of them, rank the other nodes according to the score of their label selection criterion (empirical risk for **ERM**, $f_1(v_i) - f_2(v_i)$ for **Uncertainty**), and select the top ranked nodes. The performance of **ERM** and **Uncertainty** are also averaged over 10 random selections of the initial set of labels.

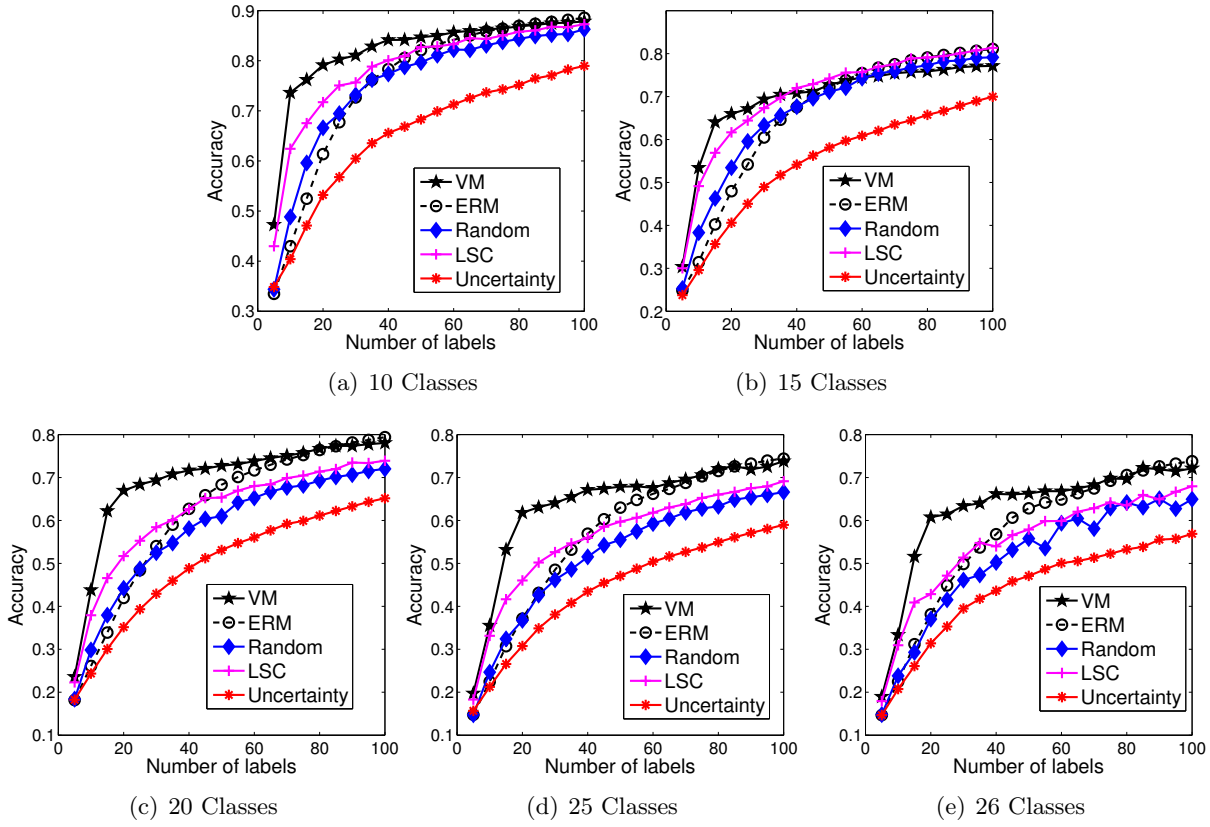
In the following, we begin with a description of the data preparation.

3.4.1 Data Preparation

Three real-world data sets are used in our experiments. The first one is the Isolet spoken letter database [1]. It contains 150 subjects who spoke the name of each letter of the alphabet twice. Hence, we have 52 examples from each speaker. The speakers are grouped into sets of 30 speakers each, and are referred to as Isolet1, Isolet2, Isolet3, Isolet4, and Isolet5. Here we use Isolet1 which contains 1560 data instances of 26 classes (spoken letters). Each class has 60 examples, and each example is represented by a 617-dimensional vector recording the spectral coefficients, contour features, sonorant features, pre-sonorant features and post-sonorant features.

The second one is the MNIST handwritten digit database [2]. This database has a training set of 60,000 images (denoted as set 1), and a testing set of 10,000 images (denoted as set 2). We take the first 1000 images from set 1 and the first 1000 images from set 2 as our experimental data. Each class (digit) contains around 200 images, each of which is of size 28×28 and therefore represented by a 784-dimensional vector.

[1] http://archive.ics.uci.edu/ml/datasets/ISOLET
[2] http://yann.lecun.com/exdb/mnist/



(a) 10 Classes (b) 15 Classes

(c) 20 Classes (d) 25 Classes (e) 26 Classes

Figure 3.1: Classification accuracy vs. the number of labels used on the Isolet data set

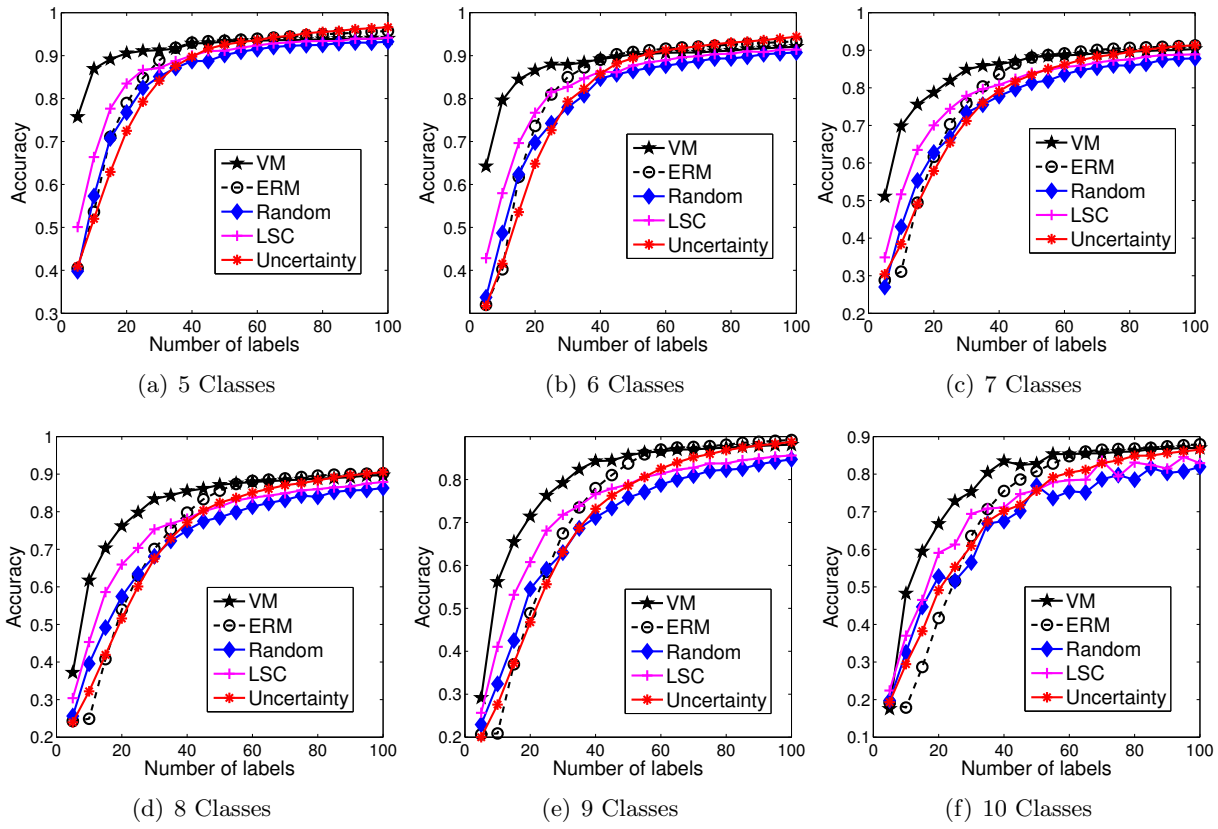
Table 3.1: Classification accuracy (%) by using 20 and 50 labels on the Isolet data set.

	10 classes		15 classes		20 classes		25 classes		26 classes		average	
# of labels	20	50	20	50	20	50	20	50	20	50	20	50
VM	**79.2**	**84.7**	**66.0**	72.7	**67.0**	**72.8**	**61.8**	**67.9**	**60.8**	**66.3**	**67.0**	**72.9**
ERM	61.4	82.1	48.0	72.2	42.0	68.3	37.1	63.0	38.2	62.8	45.3	69.7
Random	66.6	79.7	53.4	71.2	44.2	61.0	36.8	55.5	37.0	55.8	47.6	64.6
LSC	71.7	82.8	61.7	**74.2**	51.7	65.3	46.0	59.7	42.9	57.9	54.8	68.0
Uncertainty	53.2	68.3	40.6	58.1	35.2	53.1	30.8	47.1	31.4	47.1	38.2	54.7

The third data set is a connected co-author graph extracted from the DBLP database [3] on four areas: machine learning, data mining, information retrieval and database, which naturally form four classes. The co-author graph contains a total of 1711 vertices, each of which represents an author. The edge between each pair of authors is weighted by the number of papers they co-authored. Each class (research area) contains around 400 authors.

For each of the first two data sets, Isolet and MNIST, following [24], we build a 4-nearest neighbor graph among the data points, and run the active learning algorithms on graphs as well

[3] http://www.informatik.uni-trier.de/~ley/db/



(a) 5 Classes (b) 6 Classes (c) 7 Classes

(d) 8 Classes (e) 9 Classes (f) 10 Classes

Figure 3.2: Classification accuracy vs. the number of labels used on the MNIST data set

Table 3.2: Classification accuracy (%) by using 20 and 50 labels on the MNIST data set.

	5 classes		6 classes		7 classes		8 classes		9 classes		10 classes		average	
# of labels	20	50	20	50	20	50	20	50	20	50	20	50	20	50
VM	**90.6**	93.3	**86.6**	90.4	**78.8**	**88.2**	**76.2**	**87.2**	**71.4**	**85.6**	**66.8**	**82.8**	**78.4**	**87.9**
ERM	79.0	**93.5**	73.6	**90.8**	61.6	88.0	54.0	85.6	48.9	83.7	41.7	80.8	59.8	87.1
Random	76.8	90.2	69.8	86.4	62.8	81.3	57.4	78.5	54.5	75.8	52.8	77.0	62.4	81.5
LSC	83.5	91.2	76.7	87.7	70.0	84.0	65.9	81.4	60.8	78.9	59.0	75.8	69.3	83.2
Uncertainty	72.5	92.6	64.8	89.4	57.9	83.5	51.6	82.4	46.8	78.6	49.1	75.6	57.1	83.7

as the harmonic Gaussian field classifier. The third data set contains an inherent graph structure. Note that each data instance (author) in the co-author graph does not have a natural feature representation, therefore existing feature-based active learning methods cannot be directly applied to it.

3.4.2 Classification Results

For the Isolet and MNIST data sets, the experiments are conducted by choosing different numbers of classes (denoted as k) from the original data set. For Isolet, $k = 10, 15, 20, 25, 26$. For each given

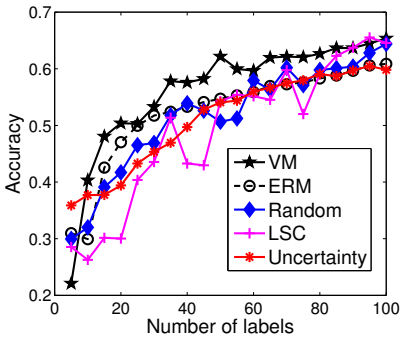


Figure 3.3: Classification accuracy vs. the number of labels used on the co-author graph.

Table 3.3: Classification accuracy (%) by using 20 and 50 labels on the co-author graph.

# of labels	20	50
VM	**50.4**	**62.2**
ERM	47.0	54.7
Random	41.7	50.7
LSC	30.0	54.3
Uncertainty	39.4	54.1

class number $k(= 10, 15, 20, 25)$, the performance scores are computed by averaging the scores of 10 repeats of different randomly chosen classes. When $k = 26$, which is the total number of classes in Isolet, we report the performance scores of using the whole data set. For each test, we employ different active learning methods to select l examples to label and train a harmonic Gaussian classifier to predict the labels of the rest of the data. Fig. 3.1 shows the plots of classification accuracy versus the number of labels used (l). For MNIST, the number of classes is chosen to be $k = 5, 6, 7, 8, 9, 10$, and we also average the classification accuracy over 10 different random selections of classes except for $k = 10$, which corresponds to using the whole data set. The classification accuracy versus the number of labels used is plotted in Fig. 3.2. For the co-author graph, since the original data set only contains four classes, we directly run experiments on the whole data set. We show the performance comparison in Fig. 3.3.

As can be observed from Fig. 3.1 to Fig. 3.3, our proposed **VM** algorithm significantly outperforms other active learning criteria on all the three data sets, especially when the number of labels is very small. **LSC** performs the second best on the Isolet and MNIST data sets when the number of labels is relatively small. It is interesting to note that on the MNIST data set, **ERM** and **Uncertainty** perform not very well when the number of labels is small, and perform much better when more labels are selected, indicating that they rely heavily on the label information of

the selected data.

We further provide the detailed classification accuracy by using 20 and 50 labels in Table 3.1∼3.3. The last two columns of Table 3.1 and Table 3.2 record the average classification accuracy over different numbers of classes. We can see that overall, **VM** performs significantly better than all the other methods, including **ERM** and **Uncertainty** that use label information. Comparing with the algorithm that performs the second best in each case, **VM** achieves 27.0% (10.6%), 29.6% (6.2%), 6.4% (16.6%) relative error reduction in the average classification accuracy using 20 (50) labels on Isolet, MNIST and the co-author graph, respectively. We have also performed the two-tailed t-tests at 95% significance level over the experimental results in Table 3.1∼3.3. In all the cases that **VM** performs the best, the p-values between the results of **VM** and other algorithms are less than 0.05. Therefore, the improvements of our proposed algorithm are statistically significant.

Chapter 4

Semi-Supervised Learning on Heterogeneous Networks: A Ranking-Based Classification Approach

4.1 Overview

Semi-supervised classification is a critical problem in the field of semi-supervised learning. On heterogeneous networked data, *semi-supervised classification* (or classification in short) and *ranking* are two of the most fundamental analytical techniques. When label information is available for some of the nodes, *classification* makes use of the labeled data as well as the network structure to predict the class membership of the unlabeled data [66, 54]. On the other hand, *ranking* gives a partial ordering to nodes in the network by evaluating the node/link properties using some ranking scheme, such as PageRank [60] or HITS [40]. Both classification and ranking have been widely studied and found to be applicable in a wide range of problems.

Traditionally, classification and ranking are regarded as orthogonal approaches, computed independently. However, adhering to such a strict dichotomy has serious downsides. Consider, for instance, an network of bibliographic data, consisting of some combination of published papers, authors, and conferences. As a concrete example, suppose we wish to classify the conferences in Table 4.1 into two research areas. We wish to minimize the chance that the top conferences are misclassified, not only to improve our classification results overall, but also because misclassifying a top conference is very likely to increase errors on many other nodes that link to that conference, and are therefore greatly influenced by its label. We would thus like to more heavily penalize classification mistakes made on highly ranked conferences, relative to a workshop of little influence. Providing a ranking of all conferences within a research area can give users a clearer understanding of that field, rather than simply grouping conferences into classes without noting their relative importance. On

Table 4.1: Conferences from two research areas

Database	SIGMOD, VLDB, ICDE, EDBT, PODS, ...
Information Retrieval	SIGIR, ECIR, CIKM, WWW, WSDM, ...

Table 4.2: Top-5 ranked conferences in different settings

Rank	Global Ranking	Within DB	Within IR
1	VLDB	VLDB	SIGIR
2	SIGIR	SIGMOD	ECIR
3	SIGMOD	ICDE	WWW
4	ICDE	PODS	CIKM
5	ECIR	EDBT	WSDM

the other hand, the class membership of each conference is very valuable for characterizing that conference. Ranking all conferences globally without considering any class information can often lead to meaningless results and apples-to-oranges comparisons. For instance, ranking database and information retrieval conferences together may not make much sense since the top conferences in these two fields cannot be reasonably compared, as shown in the second column of Table 4.2. These kinds of nonsensical ranking results are not caused by the specific ranking approach, but are rather due to the inherent incomparability between the two classes of conferences. Thus we suppose that combining classification with ranking may generate more informative results. The third and fourth columns in Table 4.2 illustrate this combined approach, showing the more meaningful conference ranking within each class.

In this study, we propose RankClass, a new framework that groups nodes into several pre-specified classes, while generating the ranking information for each type of nodes within each class simultaneously in heterogeneous networked data. More accurate classification of nodes increases the quality of the ranking within each class, since there is a higher guarantee that the ranking algorithm used will be comparing only nodes of the same class. On the other hand, better ranking scores improve the performance of the classifier, by correctly identifying which nodes are more important, and should therefore have a higher influence on the classifier's decisions. We use the ranking distribution of nodes to characterize each class, and we treat each node's label information as a

prior. By building a graph-based ranking model, different types of nodes are ranked simultaneously within each class. Based on these ranking results, we estimate the relative importance or visibility of different parts of the network with regard to each class. In order to generate better within-class ranking, the network structure employed by the ranking model is adjusted so that the sub-network composed of nodes ranked high in each specific class is emphasized, while the sub-network of the rest of the class is gradually weakened. Thus, as the network structure of each class becomes clearer, the ranking quality improves. Finally, the posterior probability of each node belonging to each class is estimated to determine each node's optimal class membership. Instead of performing ranking after classification, as facet ranking does [17, 85], RankClass essentially integrates ranking and classification, allowing both approaches to mutually enhance each other. RankClass iterates over this process until converging to a stable state. Experimental results show that RankClass both boosts the overall classification accuracy and constructs within-class rankings, which may be interpreted as meaningful summaries of each class.

4.2 Problem Formalization

In this section, we introduce several related concepts and notations, and then formally define the problem.

Definition 1. **Heterogeneous network.** Given m types of nodes, denoted by $\mathcal{X}_1 = \{x_{11}, \ldots, x_{1n_1}\}$, $\ldots, \mathcal{X}_m = \{x_{m1}, \ldots, x_{mn_m}\}$, a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$ is called a heterogeneous network if $\mathcal{V} = \bigcup_{i=1}^{m} \mathcal{X}_i$ and $m \geq 2$. $\mathcal{E}$ is the set of links between any two nodes of $\mathcal{V}$, and $\mathcal{W}$ is the set of weight values on the links. When $m = 1$, $\mathcal{G}$ reduces to a homogeneous network. ∎

In the following sections, for convenience, we use $\mathcal{X}_i$ to denote both the set of nodes belonging to the i-th type and the type name. We let $\mathcal{W}_{x_{ip}x_{jq}}$ denote the weight of the link between any two nodes x_{ip} and x_{jq}, which is represented by $\langle x_{ip}, x_{jq} \rangle$.

In a heterogeneous network, each type of link relationship between two types of nodes $\mathcal{X}_i$ and $\mathcal{X}_j$ can be represented by a relation graph $\mathcal{G}_{ij}$, $i, j \in \{1, \ldots, m\}$. Note that it is possible for $i = j$. Let $\mathbf{R}_{ij}$ be an $n_i \times n_j$ relation matrix corresponding to graph $\mathcal{G}_{ij}$. The element at the p-th row and q-th column of $\mathbf{R}_{ij}$ is denoted as $R_{ij,pq}$, representing the weight on link $\langle x_{ip}, x_{jq} \rangle$. There are many ways to define the weights on the links, which can also incorporate domain knowledge. A simple

definition is as follows:

$$R_{ij,pq} = \begin{cases} 1 & \text{if nodes } x_{ip} \text{ and } x_{jq} \text{ are linked together} \\ 0 & \text{otherwise.} \end{cases}$$

Here we consider undirected relation graphs such that $\mathbf{R}_{ij} = \mathbf{R}_{ji}^T$. In this way, each heterogeneous network $\mathcal{G}$ can be mathematically represented by a set of relation matrices $\mathcal{G} = \{\mathbf{R}_{ij}\}_{i,j=1}^m$.

To naturally generalize classification in homogeneous network data, we define a class in a heterogeneous network to be a group of multi-typed nodes sharing a common topic. For instance, a research community in a bibliographic network contains not only authors, but also papers, venues and terms belonging to the same research area. Other examples include movie networks in which movies, directors, actors and keywords are tagged with the same genre, and E-commerce networks where sellers, customers, items and tags belong to the same shopping category. The formal definition of a *class* is given below:

Definition 2. **Class.** Given a heterogeneous network $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$, $\mathcal{V} = \bigcup_{i=1}^m \mathcal{X}_i$, a class is defined as $\mathcal{G}' = \langle \mathcal{V}', \mathcal{E}', \mathcal{W}' \rangle$, where $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{E}' \subseteq \mathcal{E}$. $\forall e = \langle x_{ip}, x_{jq} \rangle \in \mathcal{E}'$, where $x_{ip} \in \mathcal{V}'$ and $x_{jq} \in \mathcal{V}'$, we have $\mathcal{W}'_{x_{ip}x_{jq}} = \mathcal{W}_{x_{ip}x_{jq}}$. Note here, $\mathcal{V}'$ also consists of multiple types of nodes from $\mathcal{X}_1$ to $\mathcal{X}_m$. ∎

Definition 2 follows [75] and [39]. Notice that a class in a heterogeneous network is actually a sub-network containing multi-typed nodes that are closely related to each other. In addition to grouping multi-typed nodes into the pre-specified K classes, we also aim to generate the ranking distribution of nodes within each class k, which can be denoted as $P(x|T(x), k)$, $k = 1, \ldots, K$. $T(x)$ denotes the type of node x. Note that different types of nodes cannot be compared in a ranking. For example, it is not meaningful to create a ranking of conferences and authors together in a bibliographic network. Therefore, each ranking distribution is restricted to a single node type, i.e., $\sum_{p=1}^{n_i} P(x_{ip}|\mathcal{X}_i, k) = 1$.

Now our problem can be formalized as follows: given a heterogeneous network $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$, a subset of nodes $\mathcal{V}' \subseteq \mathcal{V} = \bigcup_{i=1}^m \mathcal{X}_i$, which are labeled with values $\mathcal{Y}$ denoting which of the K pre-specified classes each node belongs to, predict the class labels for all the unlabeled nodes $\mathcal{V} - \mathcal{V}'$ as well as the ranking distribution of nodes within each class, $P(x|T(x), k)$, $x \in \mathcal{V}$, $k = 1, \ldots, K$.

4.3 The RankClass Algorithm

In this section we introduce our ranking-based iterative classification method, RankClass. There are two major challenges when working with heterogeneous networks: (1) how to exploit the links representing the dependency relationships between nodes; and (2) how to model the type differences among nodes and links. The intuition behind RankClass is to build a graph-based ranking model that ranks multi-typed nodes simultaneously, according to the relative importance of nodes within each class. The initial ranking distribution of each class is determined by the labeled data. During each iteration, the ranking results are used to modify the network structure to allow the ranking model to generate higher quality within-class ranking.

4.3.1 The Framework of RankClass

We first introduce the general framework of RankClass. We will explain each part of the algorithm in detail in the following subsections.

- Step 0: Initialize the ranking distribution within each class according to the labeled data, i.e., $\{P(x|T(x),k)^0\}_{k=1}^K$. Initialize the set of network structures employed in the ranking model, $\{\mathcal{G}_k^0\}_{k=1}^K$, as $\mathcal{G}_k^0 = \mathcal{G}$, $k = 1, \ldots, K$. Initialize $t = 1$.

- Step 1: Using the graph-based ranking model and the current set of network structures $\{\mathcal{G}_k^{t-1}\}_{k=1}^K$, update the ranking distribution within each class k, i.e., $\{P(x|T(x),k)^t\}_{k=1}^K$.

- Step 2: Based on $\{P(x|T(x),k)^t\}_{k=1}^K$, adjust the network structure to favor within-class ranking, i.e., $\{\mathcal{G}_k^t\}_{k=1}^K$.

- Step 3: Repeat steps 1 and 2, setting $t = t+1$ until convergence, i.e., until $\{P(x|T(x),k)^*\}_{k=1}^K = \{P(x|T(x),k)^t\}_{k=1}^K$ do not change much for all $x \in \mathcal{V}$.

- Step 4: Based on $\{P(x|T(x),k)^*\}_{k=1}^K$, calculate the posterior probability for each node, i.e., $\{P(k|x,T(x))\}_{k=1}^K$. Assign the class label to node x as:

$$C(x) = \arg\max_{1 \le k \le K} P(k|x,T(x))$$

4.3.2 Graph-based Ranking

Ranking is often used to evaluate the relative importance of nodes in a collection. In this work, we propose to rank nodes within their own type and within a specific class. The higher a node x is ranked within class k, the more important x is for class k, and the more likely it is that x will be visited in class k. Clearly, within-class ranking is quite different from global ranking, and will vary throughout different classes.

The intuitive idea of our ranking scheme is authority propagation throughout the network. Taking the bibliographic network as an example, in a specific research area, it is natural to observe the following ranking rules [75]:

1. Highly ranked conferences publish many high quality papers.
2. High quality papers are often written by highly ranked authors.
3. High quality papers often contain keywords that are highly representative of the papers' areas.

The above authority ranking rules can be summarized as follows: nodes which are linked together in a network are more likely to share similar ranking scores. Therefore, the ranking of each node can be iteratively updated by looking at the rankings of its neighbors. The initial ranking distribution within a class k can be specified by the user. When nodes are labeled without ranking information in a general classification scenario, we can initialize the ranking as a uniform distribution over only the labeled nodes:

$$P(x_{ip}|\mathcal{X}_i, k)^0 = \begin{cases} 1/l_{ik} & \text{if } x_{ip} \text{ is labeled to class } k \\ 0 & \text{otherwise.} \end{cases}$$

where l_{ik} denotes the total number of nodes of type $\mathcal{X}_i$ labeled to class k.

Suppose the current network structure used to estimate the ranking within class k is mathematically represented by the set of relation matrices: $\mathcal{G}_k^{t-1} = \{\mathbf{R}_{ij}\}_{i,j=1}^m$. For each relation matrix $\mathbf{R}_{ij}$, we define a diagonal matrix $\mathbf{D}_{ij}$ of size $n_i \times n_i$. The (p,p)-th element of $\mathbf{D}_{ij}$ is the sum of the p-th row of $\mathbf{R}_{ij}$. Instead of using the original relation matrices in the authority propagation, we

construct the normalized form of the relation matrices as follows:

$$\mathbf{S}_{ij} = \mathbf{D}_{ij}^{(-1/2)} \mathbf{R}_{ij} \mathbf{D}_{ji}^{(-1/2)}, i, j \in \{1, \ldots, m\} \tag{4.1}$$

This normalization technique is adopted in traditional graph-based learning [89] in order to reduce the impact of node popularity. In other words, we can suppress popular nodes to some extent, to keep them from completely dominating the authority propagation. Notice that the normalization is applied separately to each relation matrix corresponding to each type of link, rather than the whole network. In this way, the type differences between nodes and links are well-preserved [39]. At the t-th iteration, the ranking distribution of node x_{ip} with regard to class k is updated as follows:

$$P(x_{ip}|\mathcal{X}_i, k)^t \propto \frac{\sum_{j=1}^{m} \lambda_{ij} S_{ij,pq} P(x_{jq}|\mathcal{X}_j, k)^{t-1} + \alpha_i P(x_{ip}|\mathcal{X}_i, k)^0}{\sum_{j=1}^{m} \lambda_{ij} + \alpha_i} \tag{4.2}$$

The first term of Equation (4.2) updates the ranking score of node x_{ip} by the summation of the ranking scores of its neighbors x_{jq}, weighted by the link strength $S_{ij,pq}$. The relative importance of neighbors of different types is controlled by $\lambda_{ij} \in [0, 1]$. The larger the value of λ_{ij}, the more value is placed on the relationship between node types $\mathcal{X}_i$ and $\mathcal{X}_j$. For example, in a bibliographic network, if a user believes that the links between *authors* and *papers* are more trustworthy and influential than the links between *conferences* and *papers*, then the λ_{ij} corresponding to the *author-paper* relationship should be set larger than that of *conference-paper*. As a result, the rank of a paper will rely more on the ranks of its authors than the rank of its publication venue. The parameters λ_{ij} can also be thought of as performing feature selection in the heterogeneous network, i.e., selecting which types of links are important in the ranking process.

The second term learns from the initial ranking distribution encoded in the labels, whose contribution is weighted by $\alpha_i \in [0, 1]$. A similar strategy has been adopted in [49, 39] to control the weights between different types of relations and nodes. After each iteration, $P(x_{ip}|\mathcal{X}_i, k)^t$ is normalized such that $\sum_{p=1}^{n_i} P(x_{ip}|\mathcal{X}_i, k)^t = 1$, $\forall i = 1, \ldots, m$, $k = 1, \ldots, K$, in order to stay consistent with the mathematical definition of a ranking distribution.

We employ the authority propagation scheme in Equation (4.2) to estimate the ranking dis-

tribution instead of other simple measures computed according to the network topology (e.g., the degree of each node). This choice was made since we aim to rank nodes with regard to each class by utilizing the current soft classification results. Therefore, if the ranking of a node were merely based on the network topology, it would be the same for all classes. By learning from the label information in the graph-based authority propagation method, the ranking of each node within different classes will be computed differently, which is more suitable for our setting.

Following a similar analysis to [39] and [90], the updating scheme in Equation (4.2) can be proven to converge to the closed form solution of minimizing the following objective function:

$$\begin{aligned} J(P(x_{ip}|\mathcal{X}_i,k)) &= \sum_{i,j=1}^{m} \lambda_{ij} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} S_{ij,pq} \left(P(x_{ip}|\mathcal{X}_i,k) - P(x_{jq}|\mathcal{X}_j,k)\right)^2 \\ &\quad + \sum_{i=1}^{m} \alpha_i \sum_{p=1}^{n_i} (P(x_{ip}|\mathcal{X}_i,k) - P(x_{ip}|\mathcal{X}_i,k)^0)^2 \end{aligned} \tag{4.3}$$

which shares a similar theoretical foundation with the graph-based regularization framework on heterogeneous networks [39] that preserves consistency over each relation graph corresponding separately to each link type. However, we extend the graph-based regularization framework to rank nodes within each class, which is conceptually different from [39].

4.3.3 Adjusting the Network

Although graph-based ranking considers class information by incorporating the labeled data, it still ranks all node types in the global network. Instead, a within-class ranking should be performed over the sub-network corresponding to each specific class. The cleaner the network structure, the higher our ranking quality. Therefore, the ranking within each class should be performed over a different sub-network, rather than employing the same global network for every class. The network structure is mathematically represented by the weight values on the links. Thus, extracting the sub-network belonging to class k is equivalent to increasing the weight on the links within the corresponding sub-network, and decreasing the weight on the links in the rest of the network. It is straightforward to verify that multiplying $\mathbf{R}_{ij}$ by any positive constant c will not change the value of $\mathbf{S}_{ij}$. So increasing the weights on the links within a sub-network should be performed relative to the weight on the links of other parts of the network. In other words, we can increase or decrease

the absolute values of the weights on the links in the whole network, as long as the weights on the links of the sub-network belonging to class k are larger than those on the links belonging to the rest of the network. Let $\mathcal{G}_k^t = \{\mathbf{R}_{ij}^t(k)\}_{i,j=1}^m$. We propose a simple scheme to update the network structure so as to favor the ranking within each class k, given the current ranking distribution $P(x|T(x),k)^t$:

$$R_{ij,pq}^t(k) = R_{ij,pq} \times \left(r(t) + \sqrt{\frac{P(x_{ip}|\mathcal{X}_i,k)^t}{\max_p P(x_{ip}|\mathcal{X}_i,k)^t} \frac{P(x_{jq}|\mathcal{X}_j,k)^t}{\max_q P(x_{jq}|\mathcal{X}_j,k)^t}} \right) \tag{4.4}$$

Recall that $\mathbf{R}_{ij}$ is the relation matrix corresponding to the links between node types $\mathcal{X}_i$ and $\mathcal{X}_j$ in the original network. Using the above updating scheme, the weight of each link $\langle x_{ip}, x_{jq} \rangle$ is increased in proportion to the geometric mean of the ranking scores of x_{ip} and x_{jq}, which are scaled to the interval of $[0,1]$. The higher the rankings of x_{ip} and x_{jq}, the more important the link between them $\langle x_{ip}, x_{jq} \rangle$ is in class k. The weight on that link should therefore be increased. Note that instead of creating hard partitions of the original network into classes, we simply increase the weights on the links that are important to classes k. This is because at any time in the iteration, the current classes represented by the ranking distributions are not very accurate, and the results will be more stable if we consider both the global network structure and the current ranking results. By gently increasing the weights of links in the sub-network of class k, we gradually extract the correct sub-network from the global network, since the weights of links in the rest of the network will decrease to very low values. Note that this adjustment of the network structure still respects the differences among the various types of nodes and links.

$r(t)$ is a positive parameter that does not allow the weights of links to drop to 0 in the first several iterations, when the authority scores have not propagated very far throughout the network and $P(x|T(x),k)^t$ are close to 0 in value for many nodes. As discussed above, multiplying $\mathbf{R}_{ij}$ by any positive constant will not change the value of $\mathbf{S}_{ij}$. Therefore, it is essentially the ratio between $r(t)$ and $\sqrt{\frac{P(x_{ip}|k)^t}{\max_p P(x_{ip}|k)^t} \times \frac{P(x_{jq}|k)^t}{\max_q P(x_{jq}|k)^t}}$ that determines how much the original network structure and the current ranking distribution, respectively, contribute to the adjusted network $\mathcal{G}_k^t$. Since we hope to progressively extract the sub-network belonging to each class k, and we want to gradually reduce the weights of links that do not belong to class k down to 0, we decrease $r(t)$ exponentially by setting $r(t) = \frac{1}{2^t}$.

Equation (4.4) is not the only way to gradually increase the weights of links between highly ranked nodes in class k. For instance, the geometric mean of $\frac{P(x_{ip}|k)^t}{\max_p P(x_{ip}|k)^t}$ and $\frac{P(x_{jq}|k)^t}{\max_q P(x_{jq}|k)^t}$ can be replaced by the arithmetic mean, and $r(t)$ can be any positive function that decreases with t. We will show in Section 4.4 that even such simple adjustments as shown above can boost both the classification and ranking performance of RankClass.

4.3.4 Posterior Probability Calculation

Once the ranking distribution of each class has been computed by the iterative algorithm, we can calculate the posterior probability of each node of type $\mathcal{X}_i$ belonging to class k simply by Bayes' rule:

$$P(k|x_{ip}, \mathcal{X}_i) \propto P(x_{ip}|\mathcal{X}_i, k)P(k|\mathcal{X}_i)$$

where $P(x_{ip}|\mathcal{X}_i, k) = P(x_{ip}|\mathcal{X}_i, k)^*$, and $P(k|\mathcal{X}_i)$ represents the relative size of class k among type $\mathcal{X}_i$, which should also be estimated. We choose the $P(k|\mathcal{X}_i)$ that maximizes the likelihood of generating the set of nodes of type $\mathcal{X}_i$:

$$\log L(x_{i1}, \ldots, x_{in_i}|\mathcal{X}_i) = \sum_{p=1}^{n_i} \log P(x_{ip}|\mathcal{X}_i) = \sum_{p=1}^{n_i} \log \left(\sum_{k=1}^{K} P(x_{ip}|\mathcal{X}_i, k)P(k|\mathcal{X}_i) \right) \tag{4.5}$$

By employing the EM algorithm, $P(k|\mathcal{X}_i)$ can be iteratively estimated using the following two equations:

$$\begin{aligned} P(k|x_{ip}, \mathcal{X}_i)^t &\propto P(x_{ip}|\mathcal{X}_i, k)P(k|\mathcal{X}_i)^t \\ P(k|\mathcal{X}_i)^t &= \sum_{p=1}^{n_i} P(k|x_{ip}, \mathcal{X}_i)^t / n_i \end{aligned}$$

where $P(k|\mathcal{X}_i)$ is initialized uniformly as $P(k|\mathcal{X}_i)^0 = 1/K$.

4.3.5 Computational Complexity Analysis

In this subsection, we analyze the computational complexity of the proposed RankClass algorithm. Let K denote the number of classes, $|V|$ denote the total number of nodes, and $|E|$ denote the total number of links in the network. It takes $O(K|V|)$ time to initialize the ranking distribution in step

0. At each iteration of step 1, we need to process each link twice to update the ranking distribution, once for each node at each end of the link. We also need $O(K|V|)$ time to learn from the initial ranking distribution. So the total time complexity for step 1 at each iteration is $O(K(|E|+|V|))$. In step 2, we need $O(K|E|)$ time to adjust the network structure at each iteration. After the ranking distribution is computed, we need $O(K|V|)$ time at each iteration of the EM algorithm to calculate the posterior probability. Finally, it takes $O(K|V|)$ time to generate the final class prediction in step 4. Hence the total time complexity of the RankClass algorithm is $O\big(N_1K(|E|+|V|)+N_2K|V|\big)$, where N_1 is the number of iterations in the computation of the ranking distribution, and N_2 is the number of iterations in the EM algorithm. We will experimentally demonstrate that this algorithm converges in a few iterations. And since the number of classes K is constant, the computational complexity is generally linear in the number of links and nodes in the network.

4.4 Experiments

In this section, we apply our proposed ranking-based classification scheme, RankClass, to a real heterogeneous network extracted from the DBLP[1] database. We try to classify the bibliographic data into research communities, each of which consists of multi-typed nodes closely related to the same area. All of the experiments were conducted on a PC with 3.00GHz CPU and 8GB memory. The following five classification methods on networks are compared:

- Our proposed RankClass algorithm (RankClass).
- Graph-based regularization framework for transductive classification in heterogeneous networks (GNetMine) [39].
- Learning with Local and Global Consistency (LLGC) [89].
- Weighted-vote Relational Neighbor Classifier (wvRN) [53, 54].
- Network-only Link-based Classification (nLB) [66, 54].

LLGC is a graph-based transductive classification algorithm for homogeneous networks, while GNetMine is its extension, which works on heterogeneous networked data. Weighted-vote relational

[1] http://www.informatik.uni-trier.de/~ley/db/

neighbor classifier and link-based classification are two popular classification methods for networked data. Since a feature representation of nodes is not available for our problem, we use the network-only derivative of the link-based classifier (nLB) [54], which creates a feature vector for each node based on neighboring information. Note that LLGC, wvRN and nLB are classifiers which work with homogeneous networks, and cannot be directly applied to heterogeneous networks. In order to compare all of the above algorithms, we can transform the heterogeneous DBLP network into a homogeneous network in two ways (see Section 4.4.2): (1) disregard the type differences between nodes and treat all nodes as the same type; or (2) extract a homogeneous sub-network on one single type of nodes, if that node type is partially labeled. We try both approaches in the accuracy study. The open-source implementation of NetKit-SRL[2] [54] is employed in our experiments.

4.4.1 Data Preparation

We extracted a connected sub-network of the DBLP data set on four research areas: database, data mining, information retrieval and artificial intelligence, which naturally form four classes. As previously discussed, this heterogeneous information network is composed of four types of nodes: paper, conference, author and term. Among the four types of nodes, we have three types of link relationships: paper-conference, paper-author, and paper-term. The data set we used contains 14376 papers, 20 conferences, 14475 authors and 8920 terms, with a total number of 170794 links[3].

For accuracy evaluation, we use a labeled data set of 4057 authors, 100 papers and all 20 conferences. For more details about the labeled data set, please refer to [20, 75]. In the following sections, we randomly choose a subset of labeled nodes and use their label information in the learning process. The classification accuracy is evaluated by comparing with manually labeled results on the rest of the labeled nodes. Since terms are difficult to label even manually, as many terms may belong to multiple areas, we do not evaluate the accuracy on terms here.

[2] http://www.research.rutgers.edu/~sofmac/NetKit.html

[3] The data set is available at www.cs.illinois.edu/homes/mingji1/DBLP_four_area.zip for sharing and experiment repeatability.

Table 4.3: Comparison of classification accuracy on authors (%)

$(a\%, p\%)$ of authors and papers labeled	nLB (A-A)	nLB (A-C-P-T)	wvRN (A-A)	wvRN (A-C-P-T)	LLGC (A-A)	LLGC (A-C-P-T)	GNetMine (A-C-P-T)	RankClass (A-C-P-T)
(0.1%, 0.1%)	25.4	26.0	40.8	34.1	41.4	61.3	82.9	**85.4**
(0.2%, 0.2%)	28.3	26.0	46.0	41.2	44.7	62.2	83.4	**88.0**
(0.3%, 0.3%)	28.4	27.4	48.6	42.5	48.8	65.7	86.7	**88.5**
(0.4%, 0.4%)	30.7	26.7	46.3	45.6	48.7	66.0	87.2	**88.4**
(0.5%, 0.5%)	29.8	27.3	49.0	51.4	50.6	68.9	87.5	**89.2**
average	28.5	26.7	46.3	43.0	46.8	64.8	85.5	**87.9**

Table 4.4: Comparison of classification accuracy on papers (%)

$(a\%, p\%)$ of authors and papers labeled	nLB (P-P)	nLB (A-C-P-T)	wvRN (P-P)	wvRN (A-C-P-T)	LLGC (P-P)	LLGC (A-C-P-T)	GNetMine (A-C-P-T)	RankClass (A-C-P-T)
(0.1%, 0.1%)	49.8	31.5	62.0	42.0	67.2	62.7	**79.2**	77.7
(0.2%, 0.2%)	73.1	40.3	71.7	49.7	72.8	65.5	**83.5**	83.0
(0.3%, 0.3%)	77.9	35.4	77.9	54.3	76.8	66.6	83.2	**83.6**
(0.4%, 0.4%)	79.1	38.6	78.1	54.4	77.9	70.5	83.7	**84.7**
(0.5%, 0.5%)	80.7	39.3	77.9	53.5	79.0	73.5	84.1	**84.8**
average	72.1	37.0	73.5	50.8	74.7	67.8	82.7	**82.8**

Table 4.5: Comparison of classification accuracy on conferences (%)

$(a\%, p\%)$ of authors and papers labeled	nLB (A-C-P-T)	wvRN (A-C-P-T)	LLGC (A-C-P-T)	GNetMine (A-C-P-T)	RankClass (A-C-P-T)
(0.1%, 0.1%)	25.5	43.5	79.0	81.0	**85.0**
(0.2%, 0.2%)	22.5	56.0	83.5	85.0	**85.5**
(0.3%, 0.3%)	25.0	59.0	87.0	87.0	**90.0**
(0.4%, 0.4%)	25.0	57.0	86.5	89.5	**92.0**
(0.5%, 0.5%)	25.0	68.0	90.0	94.0	**95.0**
average	24.6	56.7	85.2	87.3	**89.5**

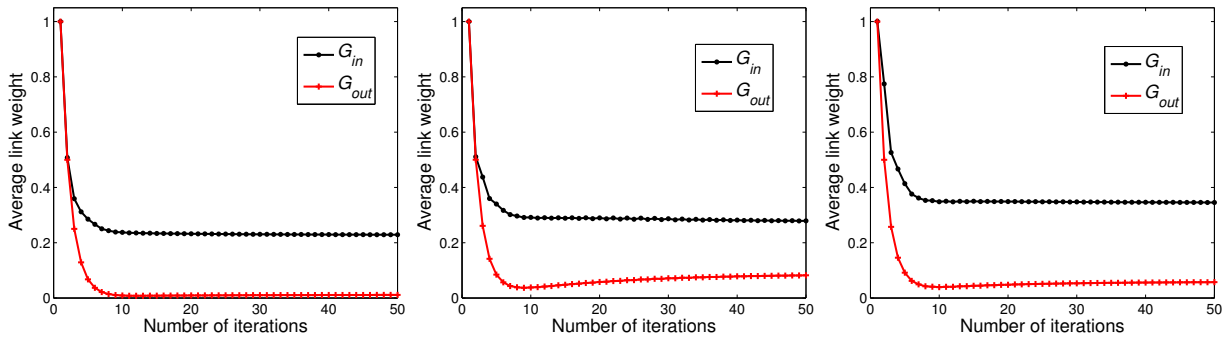
4.4.2 Accuracy Study

In order to address the label scarcity problem in real life, we randomly choose $(a\%, p\%) = [(0.1\%, 0.1\%), (0.2\%, 0.2\%), \ldots, (0.5\%, 0.5\%)]$ of authors and papers, and use their label information in the classification task. For each $(a\%, p\%)$, we average the performance scores over 10 random selections of the labeled set. We set the parameters of LLGC and GNetMine to optimal values, which were determined experimentally. For our proposed RankClass method, as discussed above, the parameters λ_{ij} are used to select which types of links are important in the ranking process. We consider all types of nodes and links to be important in the DBLP network, so we follow [39] and set $\alpha_i = 0.1$, $\lambda_{ij} = 0.2$, $\forall i, j \in \{1, \ldots, m\}$. This may not be the optimal choice, but it is good enough to demonstrate the effectiveness of our algorithm. Since labels are given for selected authors and papers, the results on conferences of wvRN, nLB and LLGC can only be obtained by mining the original heterogeneous network (denoted by A-C-P-T) and disregarding the type differences between nodes and links. While classifying authors and papers, we also tried constructing homogeneous author-author (A-A) and paper-paper (P-P) sub-networks in various ways, where the best results reported for authors are given by the co-author network, and the best results for papers are generated by linking two papers if they are published in the same conference. Note

that there is no label information given for conferences, so we cannot build a conference-conference (C-C) sub-network for classification. We show the classification accuracy on authors, papers and conferences in Tables 4.3, 4.4 and 4.5, respectively. The last row of each table records the average classification accuracy while varying the percentage of labeled data.

RankClass outperforms all other algorithms when classifying authors, papers and conferences. Note that even though the number of authors is much higher than the number of conferences, RankClass achieves comparable accuracy for both of these types of nodes. While classifying authors and papers, it is interesting to note that wvRN and nLB perform better on the *author-author* and *paper-paper* sub-networks than on the whole heterogeneous network. We observe a similar result when we use LLGC to classify papers. These results serve to verify that homogeneous classifiers like wvRN, nLB and LLGC are more suitable for working with homogeneous data. However, transforming the heterogeneous network into homogeneous sub-networks inevitably results in information loss. For example, in the *author-author* sub-network, the conferences where each author often publishes papers, and the terms that each author likes to use, are no longer known. Overall, GNetMine performs the second best by explicitly respecting the type differences in links and nodes and thus encoding the typed information in the heterogeneous network in an organized way. Compared to GNetMine, RankClass achieves 16.6%, 0.58% and 17.3% relative error reduction in the average classification accuracy when classifying authors, papers and conferences, respectively. Although RankClass has a knowledge propagation framework similar to that of GNetMine, RankClass aims to compute the within-class ranking distribution to characterize each class, and further employs the ranking results to iteratively extract the sub-network corresponding to each specific class. Therefore, the knowledge propagation for each class is more accurate.

We randomly select nodes to obtain label information in our experiments. Similar to [75], we observe that if we choose some representative (or highly ranked) nodes (e.g., famous authors) to label, the classification performance will be generally slightly better than when using labels of low quality (e.g., authors closely related to multiple fields, or with few publications). However, the difference is not significant. In other words, the initial choice of labeled data does not drastically affect the quality of ranking and classification. This is because our graph-based ranking model in Equation (4.3) is a summation of two terms, where the first depends on the initial ranking, and the



(a) Links connected to authors (b) Links connected to conferences (c) Links connected to papers

Figure 4.1: Link weight change in 50 iterations

second depends on the network structure which ensures the smoothness of the learner. Even if the quality of the initial ranking distribution is not very high, RankClass can still generate a reasonable ranking distribution and label predictions. This is because RankClass exploits the relationships among nodes in the network, iteratively propagating information throughout the network (see the first term of Equation (4.3)). Therefore, our algorithm is theoretically robust when working with random labels.

4.4.3 Convergence Study

Since our proposed RankClass algorithm iteratively adjusts the network structure to facilitate within-class ranking, we further explore the changes of the link weights within the network. According to ground truth for each class k, all of the links connected to node type $\mathcal{X}_i$ can be divided into two groups: one contains the links that connect to at least one node of class k (denoted as G_{in}), while the other group does not involve any nodes of class k (denoted as G_{out}). Links in G_{in} compose the sub-network corresponding to class k, while links in G_{out} form the sub-network excluding class k. In Figure 4.1, we show the average weight of the links in G_{in} and G_{out} (averaged over the four classes) connected to authors, conferences and papers, along with the number of iterations, when $(0.5\%, 0.5\%)$ authors and papers are labeled. The link weight changes of G_{in} and G_{out} for each individual class are very similar to Figure 4.1, and are omitted due to space limitation.

From Figure 4.1, it can be observed that the average link weight of G_{in} and G_{out} are the same at first, and then decrease at various rates over iterations. During the first several iterations, the

ranking scores have not propagated very far throughout the network and are close to 0 in value for many nodes. Therefore, the weight of many links decreases almost exponentially as a result of multiplying by $r(t)$. Then the ranking scores of nodes within each class k gradually increase, making the average link weight in G_{in} decrease more slowly than that in G_{out}. Within a few iterations, the average link weights in G_{in} and G_{out} converge to relatively stable values. There is also a clear gap between the average link weights of G_{in} and G_{out} (the former being much larger than the latter). Thus, the sub-network corresponding to each class k is well-separated from the rest of the network, and the within-class ranking can be accurately performed within the sub-network, rather than the global network. When the network structure stabilizes, the graph-based ranking scheme can be proven to converge, following a similar analysis to [89, 39].

4.4.4 Case Study

In this section, we present a simple case study by listing the top ranked nodes within each class. Recall that GNetMine performs the second best in the classification accuracy, and can generate a confidence score for each node related to each class [39]. Thus, we can also rank nodes according to the confidence scores related to each class as the within-class ranking. In Tables 4.6 and 4.7, we show the comparison of the ranking lists of conferences and terms generated by RankClass and GNetMine, respectively, with $(0.5\%, 0.5\%)$ authors and papers labeled.

From comparing the ranking lists of the two types of nodes, we can see that RankClass generates more meaningful ranking results than GNetMine. There is a high degree of consensus between the ranking list of conferences generated by RankClass and the top conferences in each research area. Similarly, the highly ranked terms generated by RankClass are in high agreement with the most representative keywords in each field. The reason why GNetMine fails to generate meaningful ranking lists is that the portions of labeled authors and papers are too limited to capture the distribution of the confidence score with regard to each class. In contrast, RankClass boosts the ranking performance by iteratively obtaining the clean sub-network corresponding to each class, which favors the within-class ranking.

Table 4.6: Top-5 conferences related to each research area generated by different algorithms

RankClass				GNetMine			
Database	Data Mining	AI	IR	Database	Data Mining	AI	IR
VLDB	KDD	IJCAI	SIGIR	VLDB	SDM	IJCAI	SIGIR
SIGMOD	SDM	AAAI	ECIR	ICDE	KDD	AAAI	ECIR
ICDE	ICDM	ICML	CIKM	SIGMOD	ICDM	ICML	CIKM
PODS	PKDD	CVPR	WWW	PODS	PAKDD	CVPR	IJCAI
EDBT	PAKDD	ECML	WSDM	CIKM	PKDD	ECML	CVPR

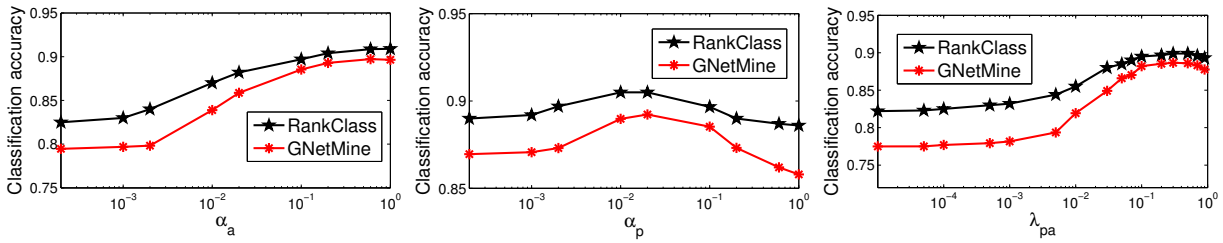
Table 4.7: Top-5 terms related to each research area generated by different algorithms

RankClass				GNetMine			
Database	Data Mining	AI	IR	Database	Data Mining	AI	IR
data	mining	learning	retrieval	interlocking	rare	failing	helps
database	data	knowledge	information	deindexing	extreme	interleaved	specificity
query	clustering	reasoning	search	seed	scan	cognition	sponsored
system	frequent	logic	web	bitemporal	mining	literals	relevance
xml	classification	model	text	debugging	associations	configuration	information

4.4.5 Model Selection

In the graph-based ranking scheme in Equation (4.2), the α_i's and λ_{ij}'s are essential parameters which control the relative importance of different types of information. In the previous experiments, we empirically set α_i's as 0.1, and λ_{ij}'s as 0.2, $\forall i, j \in \{1, \ldots, m\}$. In this subsection, we study the impact of parameters on the performance of RankClass. Since only several authors and papers are labeled, the α_i associated with authors (denoted by α_a) and papers (denoted by α_p), as well as the λ_{ij} associated with the *author-paper* relationship (denoted by λ_{pa}) are empirically more important than other parameters. Therefore we fix all other parameters and let α_a, α_p and λ_{pa} vary. As GNetMine performs the second best in the classification task, and has been proven to be robust over a large range of parameters [39], we only compare RankClass with GNetMine in this experiment. Note that we also change the corresponding parameters α_a, α_p and λ_{pa} in GNetMine. We show the average classification accuracy on three types of nodes (author, paper, conference) as a function of the parameters in Figure 4.2, with $(a\%, p\%) = (0.5\%, 0.5\%)$ authors and papers labeled.

We observe that over a large range of parameters, RankClass achieves better performance than GNetMine [39]. Since the graph-based ranking scheme in RankClass has a knowledge propagation framework similar to that of GNetMine, the changes in accuracy of the two algorithms over different parameters trend in a similar fashion. However, RankClass generates more accurate and robust results by employing the ranking results to iteratively extract the sub-network corresponding to each class. We therefore conclude that the performance of the RankClass algorithm is generally not very sensitive to the setting of its parameters.



(a) Varying α_a (b) Varying α_p (c) Varying λ_{pa}

Figure 4.2: Model Selection when (0.5%, 0.5%) of authors and papers are labeled

4.4.6 Time Complexity Study

In this section, we vary the size of the database by randomly selecting connected sub-networks from the original network, and then test the running time of our algorithm. The size of the database is measured by the number of nodes in the network. As can be seen in Figure 4.3, the time complexity of our method is generally linear with respect to the size of the database, which is consistent with our analysis in Section 4.3.5.

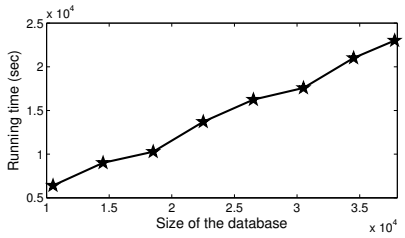


Figure 4.3: Running time w.r.t. database size

Chapter 5

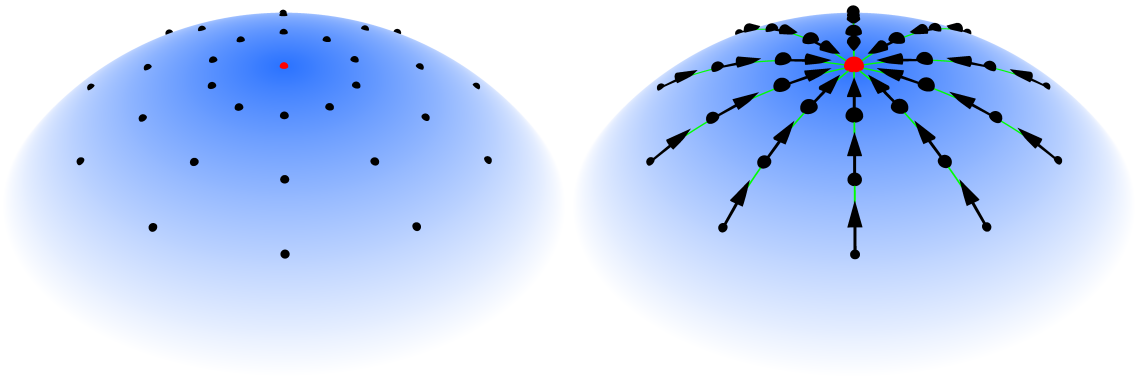
Relevance Search on Homogeneous Graphs: A Parallel Field Based Approach

5.1 Overview

Relevance search is a fundamental problem in many areas including data mining, machine learning and information retrieval [60, 40, 19]. Given a query, we aim to learn a real-valued function f that ranks the data points according to their relevance to the query. In other words, for any two data points x_i and x_j, $f(x_i) > f(x_j)$ if x_i is more relevant to the query than x_j, and vice-versa.

In many real-world homogeneous graphs, the nodes are often represented by high dimensional feature vectors, such as images, documents and videos [76, 77]. However, there is a strong intuition that the high dimensional data may have a lower dimensional intrinsic representation. Various work in literature have considered the case where the data is sampled from a submanifold embedded in the ambient Euclidean space [5, 64, 78]. In this work, we are interested in the relevance search problem on the graph nodes represented by feature vectors in Euclidean space, under the manifold assumption [90, 1, 91].

Most of the existing relevance search algorithms on data manifolds are based on the Laplacian regularization framework [90, 91], with promising performance observed on various data types such as image [29], video [84], and text [80]. These methods usually construct nearest neighbor graphs over the data to model the intrinsic geometric structure, and use the Graph Laplacian [14] to ensure that the relevance function varies smoothly along the manifold. They essentially spread the relevance scores via the graph iteratively until a stationary state is achieved. The Laplacian-based relevance search framework has some nice interpretations including close relationships to personalized PageRank [90, 60] and HITS [40]. However, recent theoretical analysis [43, 47] shows that the Laplacian regularizer is way too general for measuring the smoothness of the function. Although it is ensured that data points connected by edges in the graph have similar relevance



(a) Data samples (b) Ranking results

Figure 5.1: We aim to design a relevance function that has the highest value at the query point marked by red, and then decreases linearly along the geodesics of the manifold, which is equivalent to its gradient field being parallel along the geodesics. The arrows above denote the gradient field of the relevance function, and the green lines denote the geodesics of the data manifold.

scores, the actual variation of the relevance function along the geodesics of the data manifold is unknown. Ideally, beyond smoothness, a desirable relevance function should vary monotonically along the geodesics of the manifold. Therefore, the ranking order of the data points along the geodesics of the manifold could be well preserved. Moreover, according to the nature of the relevance search problem, no sample in the data set should be more relevant to the query than the query itself. So the relevance function should have the highest value at the query point, and then decrease to other points nearby.

In this work, we propose a novel relevance search algorithm on the data manifolds termed Parallel Field Ranking (PFRank), which learns a relevance function that has the highest value at the query point, and varies linearly and therefore monotonically along the geodesics of the data manifold. A function that varies linearly along the geodesics of the manifold is called a linear function on the manifold [62], which could still be nonlinear in the Euclidean space. It was shown that the gradient field of a linear function on the manifold has to be a parallel vector field (or parallel field in short) [62]. Besides, in order to ensure that the query has the highest relevance score, the gradient field of the relevance function should point to the query at the neighborhood around the query, since the value of a function increases towards the direction pointed by its gradient field. Figure 5.1 shows an example of the relevance function that we aim to learn on a data set sampled from a 2D sphere manifold in the 3D Euclidean space, where the red point

denotes the query, the green lines denote the geodesics, and the arrows denote the gradient field of the relevance function. The size of each point denotes the ranking order generated by the relevance function, where large points are ranked higher than small ones. As can be observed, the relevance search results generated by a function that has the highest value at the query point and varies linearly along the geodesics are quite reasonable. The gradient field of the function points to the query, and is parallel along the geodesics which pass through the query.

However, it is very difficult to design constraints on the gradient field of a function directly [47]. Instead, motivated by the recent progress in semi-supervised regression [47], we propose to learn a vector field to approximate the gradient field of the relevance function, and then design constraints on the vector field. In this work, we propose to learn a relevance function and a vector field simultaneously based on three intuitions:

1. The vector field should be close to the gradient field of the relevance function.

2. The vector field should be as parallel as possible.

3. The vector field at the neighborhood around the query should all point to the query.

By encoding the above three intuitions, the gradient field of the learned relevance function should be as parallel as possible, and is forced to point to the query at the neighborhood around the query. Hence the relevance function learned by our method decreases linearly from the query to other points along the geodesics of the data manifold.

5.2 Backgrounds

Our algorithm is fundamentally based on vector fields in geometry and vector calculus. In this section, we review some background knowledge related to vector fields.

In geometry and vector calculus, a vector field is a mapping from a manifold $\mathcal{M}$ to tangent spaces [62]. We can think of a vector field on $\mathcal{M}$ as an arrow in the same way as we think of the vector field in Euclidean space, with a given magnitude and direction, attached to each point on $\mathcal{M}$, and chosen to be tangent to $\mathcal{M}$.

As discussed before, we aim to learn a relevance function $f : \mathcal{M} \to \mathbb{R}$ that varies linearly

along the manifold $\mathcal{M}$. We first review the definitions of parallel fields and linear functions on the manifold [47].

Definition 3. **Parallel Field** [62]. A vector field X on manifold $\mathcal{M}$ is a parallel field if

$$\nabla X \equiv 0,$$

where ∇ is the covariant derivative on $\mathcal{M}$. ■

Definition 4. **Linear Function** [62]. A continuous function $f : \mathcal{M} \to \mathbb{R}$ is said to be linear if

$$(f \circ \gamma)(t) = f(\gamma(0)) + ct \tag{5.1}$$

for each geodesic γ. ■

In this work, a function f is linear means that it varies linearly along the geodesics of the manifold. This definition is a natural extension of linear functions on the Euclidean space.

Then the following proposition reveals the relationship between a parallel field and a linear function on the data manifold:

Proposition 1. [62] Let V be a parallel field on the manifold. If it is also a gradient field for function f, $V = \nabla f$, then f is a linear function on the manifold. ■

5.3 Parallel Field Ranking

In this section, we propose the objective function which learns a relevance function that decreases linearly from the query to other points along the data manifold. Let $\mathcal{M}$ be a d-dimensional submanifold in the Euclidean space $\mathbb{R}^m$. Given a graph $\mathcal{G}$ among n data points $\{x_1, \ldots, x_n\} \in \mathbb{R}^m$ on $\mathcal{M}$, where x_q is the query ($1 \leq q \leq n$), we aim to learn a relevance function $f : \mathcal{M} \to \mathbb{R}$, such that $\forall i, j \in \{1, \ldots, n\}$, $f(x_i) > f(x_j)$ if x_i is more relevant to the query x_q than x_j, and vice-versa.

5.3.1 Ensuring the Linearity

As discussed before, we aim to design regularization terms that ensure the linearity of the relevance function with respect to the data manifold, which is equivalent to ensuring the parallelism of the

gradient field of the function. However, it is very difficult to design constraints on the gradient field of a function directly [47]. Following the above analysis, we propose to learn a vector field to approximate the gradient field of the relevance function, and require the vector field to be as parallel as possible. Let $C^\infty(\mathcal{M})$ denote smooth functions on $\mathcal{M}$. Following [47], we learn a relevance function f and a vector field V on the manifold simultaneously with two constraints, which correspond to the first two intuitions proposed in Section 5.1:

- The vector field V should be close to the gradient field ∇f of f, which can be formularized as follows:

$$\min_{f \in C^\infty, V} R_1(f, V) = \int_{\mathcal{M}} ||\nabla f - V||^2 \tag{5.2}$$

- The vector field V should be as parallel as possible:

$$\min_{V} R_2(V) = \int_{\mathcal{M}} ||\nabla V||_F^2 \tag{5.3}$$

  where ∇ is the covariant derivative on the manifold, and $||\cdot||_F$ denotes the Frobenius norm.

∇V measures the change of the vector field V. If ∇V vanishes, then V is a parallel field. Then the following objective function learns a f that varies linearly along the manifold, from the vector field perspective [47]:

$$\begin{aligned} & \arg\min_{f \in C^\infty(\mathcal{M}), V} E(f, V) \\ = \; & R_0(x_q, y_q, f) + \lambda_1 R_1(f, V) + \lambda_2 R_2(V) \end{aligned} \tag{5.4}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are two regularization parameters. R_0 is a loss function that ensures the predicted relevance score for the query x_q to be close to a constant positive number y_q. If we remove R_0 from Eq. (5.4), then if f is an optimal solution to $\arg\min_{f \in C^\infty(\mathcal{M}), V} \lambda_1 R_1(f, V) + \lambda_2 R_2(V)$, it is easy to check that $f + c$ is also an optimal solution, where c could be any constant number. Therefore, by adding the R_0 term, we can get a unique solution of our relevance function f. In principle, y_q could be any positive number. Following [90, 47], we set $y_q = 1$, and use the squared loss $R_0(x_q, y_q, f) = (f(x_q) - y_q)^2$ for simplicity.

5.3.2 Discretization

Since the manifold $\mathcal{M}$ is unknown, the function f which minimizes Eq. (5.4) can not be directly solved. Following [47], we introduce how to discretize the continuous objective function (5.4) in this subsection.

For simplicity, let $f_i = f(x_i)$, $i = 1, \ldots, n$. Then our goal is to learn a relevance score vector $f = [f_1, \ldots, f_n]^T$. Let W be the corresponding affinity matrix of the graph $\mathcal{G}$ among the data examples. Then for each x_i, we can estimate its tangent space $T_{x_i}\mathcal{M}$ by performing PCA on its local neighborhood. We choose the eigenvectors corresponding to the d largest eigenvalues since $T_{x_i}\mathcal{M}$ is d-dimensional. Let $T_i \in \mathbb{R}^{m\times d}$ be the matrix whose columns constitute an orthonormal basis for $T_{x_i}\mathcal{M}$. It is easy to show that $P_i = T_i T_i^T$ is the *unique* orthogonal projection from $\mathbb{R}^m$ onto the tangent space $T_{x_i}\mathcal{M}$ [21]. That is, for any vector $a \in \mathbb{R}^m$, we have $P_i a \in T_{x_i}\mathcal{M}$ and $(a - P_i a) \perp P_i a$.

Let V be a vector field on the manifold $\mathcal{M}$. For each point x_i, let V_{x_i} denote the value of the vector field V at x_i, and $\nabla V|_{x_i}$ denote the value of ∇V at x_i. According to the definition of vector field, V_{x_i} should be a vector in tangent space $T_{x_i}\mathcal{M}$. Therefore, it can be represented by the local coordinates of the tangent space, $V_{x_i} = T_i v_i$, where $v_i \in \mathbb{R}^d$. We define $\mathbb{V} = \left[v_1^T, \ldots, v_n^T\right]^T \in \mathbb{R}^{dn}$. In other words, $\mathbb{V}$ is a dn-dimensional column vector concatenating all the v_i's.

Then according to the analysis in [47], R_1 reduces to the following:

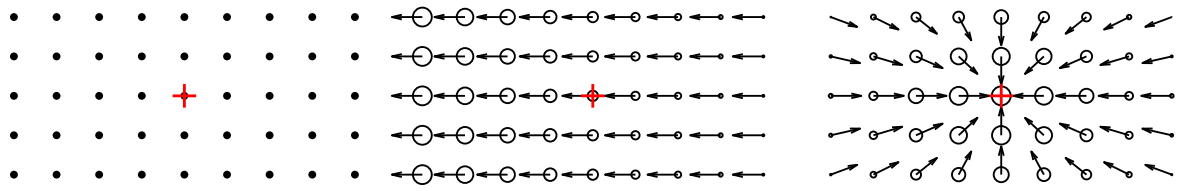
$$R_1(f, \mathbb{V}) = \sum_{i,j=1}^{n} w_{ij} \left((x_j - x_i)^T T_i v_i - f_j + f_i\right)^2 \tag{5.5}$$

And R_2 can be discretized to the following:

$$R_2(\mathbb{V}) = \sum_{i,j=1}^{n} w_{ij} \left|\left|P_i T_j v_j - T_i v_i\right|\right|^2 \tag{5.6}$$

5.3.3 Objective Function in the Discrete Form

As discussed in Section 5.1, a function that varies linearly along the manifold is good for relevance search. However, merely ensuring the linearity is not enough. Furthermore, we hope that the relevance function has the highest value at the query point, i.e., no sample in our data set is more



(a) Data samples (b) Ranking results without R_3 (c) Ranking results with R_3

Figure 5.2: A toy example explaining the reason of adding R_3.

relevant to the query than the query itself. Therefore, the relevance function should decrease from the query to neighboring data points, which is equivalent to that the gradient field of the relevance function at the neighborhood around the query should all point to the query. Figure 5.2 presents a simple toy problem to illustrate this idea. Suppose we are given a set of data points sampled from a 2D rectangle as shown in Figure 5.2(a). The query is marked by '+'. Figure 5.2(b) shows the relevance search results without employing a third regularizer R_3 which addresses our third intuition, where the arrows denote the gradient field of the relevance function, and the size of the circle at each point denotes the ranking order. Points marked by large circles are ranked higher than those marked by small circles. It is easy to see that the relevance function learned in Figure 5.2(b) varies linearly, and its gradient field is a parallel field. However, some data points are even ranked higher than the query itself, and points far from the query could be ranked higher than those close to the query, which is not reasonable. By addressing the third intuition via R_3, we force the gradient field at the neighborhood around the query to point to the query while still requiring the gradient field to be parallel along the geodesics, generating good ranking results as shown in Figure 5.2(c).

Since we approximate the gradient field of the relevance function by the vector field V through R_1, we can easily control the gradient field using V. Therefore, we require V at the neighborhood around the query to point to the query. Let $i \sim j$ denote that x_i and x_j are neighbors. We now propose the third regularization term R_3, which addresses the third intuition proposed in Section

5.1:

$$\begin{aligned} R_3(\mathbb{V}) &= \sum_{j\sim q} ||V_{x_j} - P_j(x_q - x_j)||^2 \\ &= \sum_{j\sim q} ||T_j v_j - P_j(x_q - x_j)||^2 \end{aligned} \tag{5.7}$$

Recall that V_{x_j} denotes the value of the vector field V at x_j, which is a vector in tangent space $T_{x_j}\mathcal{M}$. $(x_q - x_j)$ is a vector pointing from a neighboring point x_j to x_q, and $P_j(x_q - x_j)$ is its projection to $T_{x_j}\mathcal{M}$. Therefore, ensuing V_{x_j} and $P_j(x_q - x_j)$ to be similar forces the vector field at x_j to point from x_j to x_q.

Let $\mathbb{I}$ denote a $n \times n$ matrix where the entry at the q-th row and q-th column is 1, and all the other entries are 0. Let $y \in \mathbb{R}^n$ be a column vector where the q-th entry is $y_q(= 1)$, and all the other entries are 0. Then we have:

$$R_0(x_q, y_q, f) = (f - y)^T \mathbb{I}(f - y) \tag{5.8}$$

Combining R_0 in Eq. (5.8), R_1 in Eq. (5.5), R_2 in Eq. (5.6) and R_3 in Eq. (5.7), our final objective function can be formulated as follows:

$$\begin{aligned} J(f, \mathbb{V}) &= R_0(x_q, y_q, f) + \lambda_1 R_1(f, \mathbb{V}) + \lambda_2 R_2(\mathbb{V}) + \lambda_3 R_3(\mathbb{V}) \\ &= (f - y)^T \mathbb{I}(f - y) \\ &\quad + \lambda_1 \sum_{i,j=1}^{n} w_{ij} \left((x_j - x_i)^T T_i v_i - f_j + f_i\right)^2 \\ &\quad + \lambda_2 \sum_{i,j=1}^{n} w_{ij} \left|\left|P_i T_j v_j - T_i v_i\right|\right|^2 \\ &\quad + \lambda_3 \sum_{j\sim q} ||T_j v_j - P_j(x_q - x_j)||^2 \end{aligned} \tag{5.9}$$

The trade-off among the three regularization terms is controlled by the parameters λ_1, λ_2 and λ_3 in the range of $(0, +\infty)$.

5.4 Optimization

In this section, we discuss how to solve our objective function (5.9).

Let D be a diagonal matrix where $D_{ii} = \sum_{j=1}^{n} w_{ij}$. Let $L = D - W$ denote the Laplacian matrix [14] of the graph. Then we can rewrite R_1 as follows:

$$\begin{aligned} R_1(f, \mathbb{V}) &= 2f^T L f + \sum_{i,j=1}^{n} w_{ij} \left((x_j - x_i)^T T_i v_i\right)^2 \\ &\quad -2 \sum_{i,j=1}^{n} w_{ij} (x_j - x_i)^T T_i v_i s_{ij}^T f \end{aligned} \tag{5.10}$$

where $s_{ij} \in \mathbb{R}^n$ is a selection vector of all zero elements except for the i-th element being -1 and the j-th element being 1. We further construct a $dn \times dn$ block diagonal matrix G, and a $dn \times n$ block matrix $C = [C_1^T, \ldots, C_n^T]^T$. Let G_{ii} denote the i-th $d \times d$ diagonal block of G, and C_i denote the i-th $d \times n$ block of C, we define:

$$G_{ii} = \sum_{j \sim i}^{n} w_{ij} T_i^T (x_j - x_i)(x_j - x_i)^T T_i \tag{5.11}$$

$$C_i = \sum_{j \sim i}^{n} w_{ij} T_i^T (x_j - x_i) s_{ij}^T \tag{5.12}$$

With some algebraic transformations, it is easy to check that R_1 in Eq. (5.10) can be written as follows:

$$R_1(f, \mathbb{V}) = 2f^T L f + \mathbb{V}^T G \mathbb{V} - 2\mathbb{V}^T C f \tag{5.13}$$

Similarly, in order to rewrite R_2 into a simplified form, we define $Q_{ij} = T_i^T T_j$, $i, j = 1, \ldots, n$. Let I denote the identity matrix of size $n \times n$. We further construct a $dn \times dn$ sparse block matrix B. Let B_{ij} denote each $d \times d$ block, $i, j = 1, \ldots, n$, then we define

$$B_{ii} = \sum_{j \sim i} w_{ij} (Q_{ij} Q_{ij}^T + I) \tag{5.14}$$

$$B_{ij} = -2 w_{ij} Q_{ij} \tag{5.15}$$

Then we can rewrite R_2 as follows:

$$R_2(\mathbb{V}) = \mathbb{V}^T B \mathbb{V} \tag{5.16}$$

For R_3, we construct a $dn \times dn$ block diagonal matrix D, and a $dn \times 1$ block vector $H = [H_1^T, \ldots, H_n^T]^T$. Let D_{jj} denote the j-th $d \times d$ diagonal block of D, and H_j denote the j-th $d \times 1$ block of H. We define:

$$D_{jj} = \begin{cases} I_d, & \text{if } j \sim q \\ 0, & \text{otherwise} \end{cases} \tag{5.17}$$

$$H_j = \begin{cases} T_j^T(x_q - x_j), & \text{if } j \sim q \\ 0, & \text{otherwise} \end{cases} \tag{5.18}$$

where I_d is an identity matrix of size $d \times d$. Now we can rewrite R_3 as follows:

$$R_3(\mathbb{V}) = \mathbb{V}^T D \mathbb{V} - 2H^T \mathbb{V} + \sum_{j \sim q} ||P_j(x_q - x_j)||^2 \tag{5.19}$$

Combining R_1 in Eq. (5.13), R_2 in Eq. (5.16) and R_3 in Eq. (5.19), we get the following simplified form of our objective function:

$$\begin{aligned} & J(f, \mathbb{V}) \\ = \; & (f - y)^T \mathbb{I} (f - y) \\ & + \lambda_1 (2 f^T L f + \mathbb{V}^T G \mathbb{V} - 2 \mathbb{V}^T C f) \\ & + \lambda_2 \mathbb{V}^T B \mathbb{V} \\ & + \lambda_3 (\mathbb{V}^T D \mathbb{V} - 2 H^T \mathbb{V} + \sum_{j \sim q} ||P_j(x_q - x_j)||^2) \\ = \; & f^T (\mathbb{I} + 2\lambda_1 L) f - 2 y^T \mathbb{I} f - 2 \lambda_1 \mathbb{V}^T C f \\ & + \mathbb{V}^T (\lambda_1 G + \lambda_2 B + \lambda_3 D) \mathbb{V} - 2 \lambda_3 H^T \mathbb{V} \\ & + y^T \mathbb{I} y + \lambda_3 \sum_{j \sim q} ||P_j(x_q - x_j)||^2 \end{aligned} \tag{5.20}$$

From Eq. (5.20), it is easy to compute the partial derivative of $J(f, \mathbb{V})$ with respect to f and $\mathbb{V}$ as follows:

$$\frac{\partial J(f,\mathbb{V})}{\partial f} = 2(\mathbb{I} + 2\lambda_1 L)f - 2y - 2\lambda_1 C^T \mathbb{V} \tag{5.21}$$

$$\frac{\partial J(f,\mathbb{V})}{\partial \mathbb{V}} = -2\lambda_1 Cf + 2(\lambda_1 G + \lambda_2 B + \lambda_3 D)\mathbb{V} - 2\lambda_3 H \tag{5.22}$$

Requiring that the derivatives vanish, our solution could be obtained by solving the following linear equation system:

$$(\mathbb{I} + 2\lambda_1 L)f - \lambda_1 C^T \mathbb{V} = y \tag{5.23}$$

$$-\lambda_1 Cf + (\lambda_1 G + \lambda_2 B + \lambda_3 D)\mathbb{V} = \lambda_3 H \tag{5.24}$$

which is further equivalent to solving the following linear system:

$$\begin{pmatrix} \mathbb{I} + 2\lambda_1 L & -\lambda_1 C^T \\ -\lambda_1 C & \lambda_1 G + \lambda_2 B + \lambda_3 D \end{pmatrix} \begin{pmatrix} f \\ \mathbb{V} \end{pmatrix} = \begin{pmatrix} y \\ \lambda_3 H \end{pmatrix} \tag{5.25}$$

5.5 Computational Complexity Analysis

The computational complexity of our proposed Parallel Field Ranking (PFRank) algorithm is dominated by three parts: searching for k nearest neighbors, computing the local tangent space and solving a sparse linear system. For the k nearest neighbor search, the complexity is $O((m+k)n^2)$, where mn^2 is the complexity of computing the distance between any two data points, and kn^2 is the complexity of finding the k nearest neighbors for all the data points. The complexity for local PCA is $O(mk^2)$. Therefore, the complexity for computing the local tangent space for all the data points is $O(mnk^2)$. Solving the final sparse linear system has complexity $O(kd^2n)$. In this way, the total computational complexity for our PFRank method is $O((m+k)n^2 + mnk^2 + knd^2)$. Empirically, d and k are usually small constants less than 10. So the total computational complexity could be $O(mn^2)$.

Based on the above analysis, there are two possible ways to improve the efficiency of the last two parts (computing the local tangent space and solving a sparse linear system). Regarding to

the tangent spaces computation, we first notice that the tangent spaces varies smoothly as long as the data manifold is smooth. Therefore, for large data sets, it is not necessary to do SVD for every data point. Instead, we can select a small subset of data points and then compute local tangent spaces for this subset. And then we may apply semi-supervised learning methods to the other data points to obtain the tangent spaces. Regarding to the linear system computation, we can use an iterative way to compute the "parallel" vector field rather then solving a large sparse linear system. It is worth noticing that we only require the vector field to be parallel along the geodesics passing through the query. And it is known that the tangent vectors near the query should point to the query. Therefore, we can learn the vector field via propagating parallel tangent vectors from the neighborhood of the query to other regions. It makes sense as the learned vector field will have high accuracy near the query, which is crucial for the ranking problem.

In the case that the query data point is out of the database, we can first select a few anchor points from the database, such as the cluster centers, to help model the data. Then we can quickly compute the relationship between the query and the several anchor points to update the nearest neighbor graph and compute the local tangent space in a dynamic way following the philosophy in [81].

5.6 Experiments

In this section, we empirically evaluate the effectiveness of our proposed Parallel Field Ranking (PFRank) algorithm with comparison to several existing relevance search algorithms. Since we focus on learning a relevance function under the manifold assumption, the main comparative method is the Laplacian-based manifold ranking algorithm (MR) [90]. We construct the same nearest neighbor graph and empirically set the number of nearest neighbors to be 15 for both PFRank and MR. The parameter setting of MR is the same as the original paper [90, 29]. We empirically set $\lambda_1 = \lambda_2 = \lambda_3 = 0.01$ in our PFRank algorithm.

We also compare with the SVM method, which has been successfully applied to image retrieval [79]. With the specified relevant/irrelavant information, a maximal margin hyperplane could be built to separate the relevant data points from the irrelevant ones. Then we can return the data points farthest from the SVM boundary as the relevant ones. We use the LIBSVM toolbox [12] in

our experiments. However, with a single query, there are no irrelevant data specified by the user. Following [81], we adopt the pseudo relevance feedback strategy [55]. Specifically, the nearest 10 data points to the query measured by the Euclidean distance are considered to be relevant, and the farthest 10 are treated as irrelevant. Then we can run SVM for relevance search.

In the following, we begin with a simple synthetic example to give some intuition about how PFRank works.

5.6.1 Synthetic Example

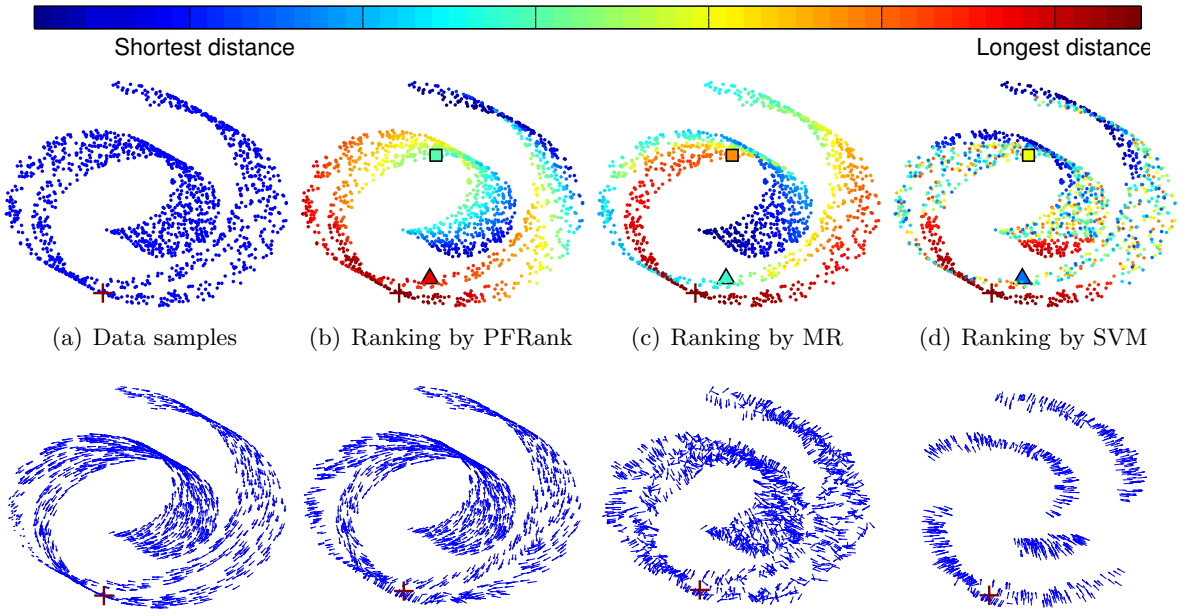
A simple synthetic example is given in Figure 5.3. We randomly sample 1500 data points from a Swiss roll with a hole as shown in Figure 5.3(a). This data set lies on a 2D manifold in the 3D Euclidean space. The query is marked by '+'.

Figure 5.3(b)∼(d) show the ranking order generated by PFRank, MR and SVM, respectively, where the data points marked by warmer color (such as red) are ranked higher than those marked by colder color (such as blue). We can see that SVM does not take the manifold structure of the data into consideration. The relevance functions generated by both PFRank and MR vary smoothly along the data manifold, and PFRank better preserves the ranking order of the data points along the geodesics of the manifold. For example, the geodesic distance between the query and the triangle point is smaller than that between the query and the square point. Therefore, the triangle point should rank higher than the square point. However, MR ranks the square point higher than the triangle point, which is counterintuitive.

Note that once the relevance function f is obtained, we can estimate its gradient field reversely. Specifically, the gradient field at each point x_i $(1 \leq i \leq n)$, denoted by $\nabla f|_{x_i}$, can be computed by minimizing the following objective function at the local neighborhood of x_i:

$$\nabla f|_{x_i} = \arg\min_{g \in \mathbb{R}^m} \sum_{j \sim i} \left(f(x_j) - f(x_i) - g^T P_i \left(x_j - x_i \right) \right)^2 \tag{5.26}$$

We further plot the vector field learned by PFRank in Figure 5.3(e), and plot the gradient fields of the relevance functions learned by PFRank, MR and SVM (computed via Eq. (5.26)) in Figure 5.3(f)∼(h), respectively. For SVM, many data points have the same relevance scores (i.e.,



(a) Data samples (b) Ranking by PFRank (c) Ranking by MR (d) Ranking by SVM

(e) Vector field by PFRank (f) Gradient field by PFRank (g) Gradient field by MR (h) Gradient field by SVM

Figure 5.3: Performance comparison of various relevance search algorithms on a toy data set. (a) shows the data set sampled from a Swiss roll with a hole, where the query is denote by '+'. (b) shows the ranking order generated by PFRank. We can see that PFRank successfully preserves the ranking order of the data points along the geodesics of the manifold. For example, the geodesic distance between the query and the point marked by '▲' is smaller than that between the query and the point marked by '■'. So '▲' is ranked higher than '■'. (c) shows the ranking order generated by MR. We can see that '▲' is ranked lower than '■', which is counterintuitive. (d) shows the ranking order generated by SVM, which does not take the manifold structure into consideration. (e) and (f) show the vector field and the gradient field of the relevance function learned by PFRank, respectively, which are quite parallel and point to the query. (g) and (h) show the gradient fields of the relevance functions learned by MR and SVM, respectively, which do not vary smoothly along the manifold.

same distance to the SVM boundary), therefore are ranked randomly. And the gradient field of the relevance function has 0 values at many places, therefore are left blank in Figure 5.3(h). Note that the value of a relevance function increases towards the direction pointed by its gradient field. Therefore, the ranking orders of data points get higher towards the direction pointed by the arrows of the gradient field. It can be observed that both the vector field and the gradient field of the relevance function learned by PFRank point to the query along the geodesics of the manifold, and are quite parallel. On the contrary, the gradient fields of the relevance functions learned by MR and SVM do not vary smoothly along the manifold, and therefore cannot preserve the ranking

order.

5.6.2 Image Retrieval

In this subsection, we apply our proposed relevance search algorithm to the image retrieval problem in real world image databases. Given an image as a query, we hope to rank the images in our database according to their relevance to the query. We begin with a description of the data preparation.

Data Preparation

Three real world data sets are used in our experiments. The first one contains 5,000 images of 50 semantic categories, from the COREL database. For each image, we extract a 297-dimensional feature vector which combines the following information:

- Grid Color Moment: Each image is partitioned into 3×3 grids. For each grid, there color moments: mean, variance and skewness are extracted in each color channel (R, G, and B) respectively. Thus, an 81-dimensional grid color moment vector is adopted.

- Edge: The Canny edge detector [11] is used to obtain the edge map for the edge orientation histogram, which is quantized into 36 bins of 10 degrees each. An additional bin is to count the number of pixels without edge information. Hence, a 37-dimensional vector is used.

- Gabor Wavelets Texture: Each image is first scaled to 64× 64 pixels. The Gabor wavelet transform [42] is then applied on the scaled image with 5 levels and 8 orientations, which results in 40 subimages. For each subimage, 3 moments are calculated: mean, variance and skewness. Thus, a 120-dimensional vector is used.

- Local Binary Pattern: The LBP [59] is a gray-scale texture measure derived from a general texture definition in a local neighborhood. A 59-dimensional LBP histogram vector is adopted.

The second data set is from the CMU PIE face database [72]. This database contains 68 subjects with 41,368 face images as a whole. The face images were captured by 13 synchronized cameras and 21 flashes, under varying pose, illumination and expression. In this experiment, we choose

the frontal pose (C27) with varying lighting conditions, which leaves us 21 images per subject. Preprocessing to locate the faces were applied. Original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then the facial areas were cropped into the final image for matching. The size of each cropped image in all the experiments is 32×32 pixels, with 256 gray levels per pixel. Therefore, each image can be represented by a 1024-dimensional feature vector in the image space. No further preprocessing is done.

The third data set is from the the Extended Yale-B database[1]. This database contains 16128 images of 38 human subjects under 9 poses and 64 illumination conditions. In this experiment, we choose the frontal pose and use all the images under different illumination. Finally we get 2414 images in total. All the face images are manually aligned and cropped, where the size of each cropped image is 32×32 pixels, with 256 gray levels per pixel. Thus each image is also represented by a 1024-dimensional vector.

Experimental Settings

We describe our experimental setup in this subsection. Both of the two image data sets we use have category labels. In the COREL data set, images from the same category belong to the same semantic concept, such as eagle, bird, elephant, etc. In CMU PIE data set, images from the same category belong to the same person (subject). Therefore, given a query, images that belong to the same category as the query are judged relevant. For each data set, we randomly choose 10 images from each category as queries, and average the retrieval performance over all the queries. It is worth noticing that both PFRank and MR need to invert a weight matrix corresponding to the data graph whose size is at least $n \times n$, which is time consuming. In order to make the relevance search scheme more efficient, given a query image, we first rank all the images according to the Euclidean distance to the query. Then we choose the top 500 images as candidates and use different relevance search algorithms to re-rank them.

We use precision, recall, Mean Average Precision (MAP) [55] and Normalized Discount Cumulative Gain (NDCG) to evaluate the relevance search results of different algorithms. Precision is defined as the number of relevant presented images divided by the number of presented images.

[1] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html

Recall is defined as the number of relevant presented images divided by the total number of relevant images in our database. Given a query, let r_i be the relevance score of the image ranked at position i, where $r_i = 1$ if the image is relevant to the query and $r_i = 0$ otherwise. Then we can compute the Average Precision (AP):

$$\text{AP} = \frac{\sum_i r_i \times \text{Precision@}i}{\#\text{ of relevant images}} \tag{5.27}$$

MAP is the average of AP over all the queries. And NDCG at position n is defined as:

$$\text{NDCG@}n = Z_n \sum_{i=1}^{n} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{5.28}$$

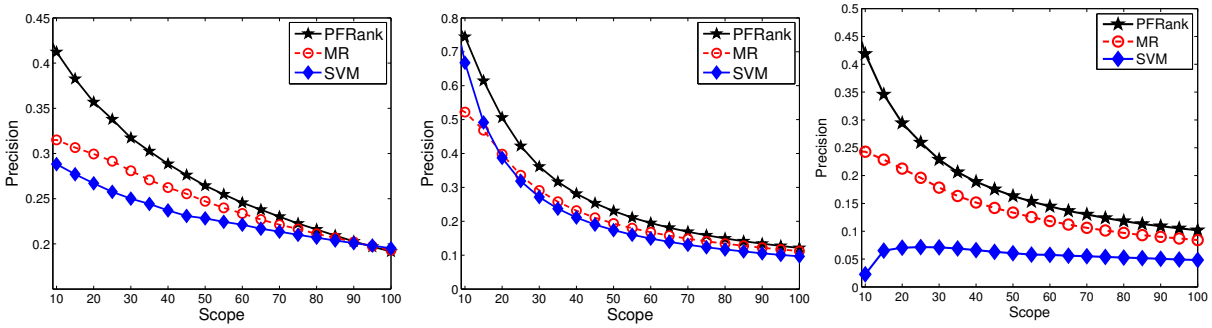
n is also called the scope, which means the number of top-ranked images presented to the user. Z_n is chosen such that the perfect ranking has a NDCG value of 1.

For our PFRank algorithm, the dimensionality of the manifold (d) in the real data is unknown. We perform cross-validation and choose $d = 2$ for the COREL data set, and choose $d = 9$ for the CMU PIE data set, respectively. All the other parameter settings are the same as in the previous experiment.

Performance Evaluation

Figure 5.4 shows the average precision-scope curves of various methods on the three image data sets, respectively. The precision-scope curve describes the precision with various scopes, and therefore providing an overall performance evaluation of the algorithms. As can be seen, our proposed PFRank algorithm consistently outperforms the other two algorithms on all the three data sets. MR generally achieves higher precision than SVM, indicating that considering the manifold structure in the data is useful for image retrieval. Note that SVM performs much worse than MR and PFRank in the Yale-B data set, which is probably because this data set has particularly good manifold structure, and SVM does not consider that.

In order to have a comprehensive view of the ranking performance, we present the NDCG, recall and MAP scores of different algorithms on the three image data sets in Table 5.1, Table 5.2 and Table 5.3, respectively. Overall, our PFRank method performs the best on all the three data sets. Although the precision of MR is generally higher than SVM in Figure 5.4, the NDCG scores



(a) COREL (b) CMU PIE (c) Yale-B

Figure 5.4: The average precision-scope curves of different algorithms on three image data sets.

Table 5.1: Performance evaluated by different metrics on the COREL data set

	NDCG@10	NDCG @20	Recall@10	Recall@20	Recall@50	MAP
SVM	0.297	0.279	0.066	0.121	0.242	0.244
MR	0.312	0.302	0.061	0.116	0.237	0.237
PFRank	**0.435**	**0.388**	**0.091**	**0.152**	**0.272**	**0.256**

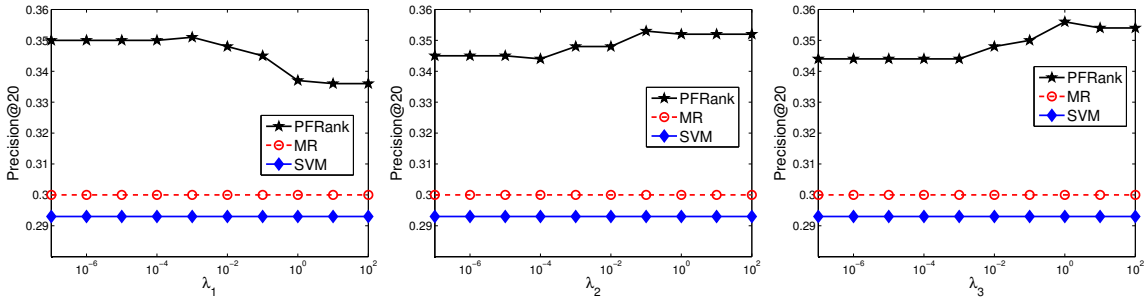
Table 5.2: Performance evaluated by different metrics on the CMU PIE data set

	NDCG@10	NDCG @20	Recall@10	Recall@20	Recall@50	MAP
SVM	0.761	0.530	0.503	0.570	0.631	0.575
MR	0.537	0.446	0.383	0.583	0.698	0.441
PFRank	**0.786**	**0.605**	**0.572**	**0.750**	**0.847**	**0.729**

Table 5.3: Performance evaluated by different metrics on the Yale-B data set

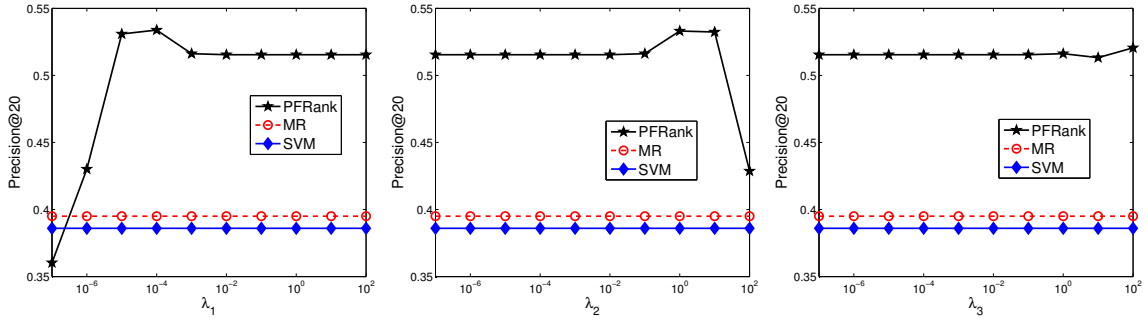
	NDCG@10	NDCG @20	Recall@10	Recall@20	Recall@50	MAP
SVM	0.024	0.058	0.014	0.074	0.157	0.054
MR	0.242	0.222	0.113	0.200	0.333	0.184
PFRank	**0.471**	**0.365**	**0.237**	**0.329**	**0.454**	**0.315**

of MR are lower than that of SVM on the CMU PIE data set. This is because the precision@n of MR is lower than that of SVM when $n < 20$, and NDCG computes the cumulative relevance scores of the top ranked images. Moreover, the recall and MAP of MR are lower than SVM on the COREL data set. MAP provides a single figure measure of quality across recall levels. Our PFRank achieves the highest MAP on all the three data sets, indicating reliable performance over the entire ranking list.



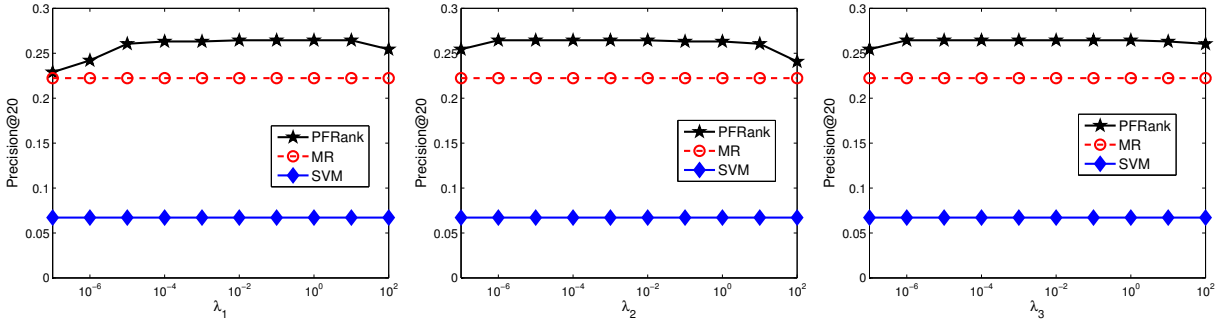
(a) Varying λ_1 (b) Varying λ_2 (c) Varying λ_3

Figure 5.5: Model selection on the COREL data set.



(a) Varying λ_1 (b) Varying λ_2 (c) Varying λ_3

Figure 5.6: Model selection on the CMU PIE data set.



(a) Varying λ_1 (b) Varying λ_2 (c) Varying λ_3

Figure 5.7: Model selection on the Yale-B data set.

Model Selection

Model selection is a critical problem in most of the learning problems. In some situations, the learning performance may drastically vary with different choices of the parameters. λ_1, λ_2 and λ_3 are essential parameters in PFRank which control the trade-off among the three regularization terms. We empirically set $\lambda_1 = \lambda_2 = \lambda_3 = 0.01$ in all the previous experiments. In this subsection, we try to study the impact of the parameters on the performance of our PFRank algorithm. Specifically, we fix other parameters the same as before, and let one of $\{\lambda_1, \lambda_2, \lambda_3\}$ vary.

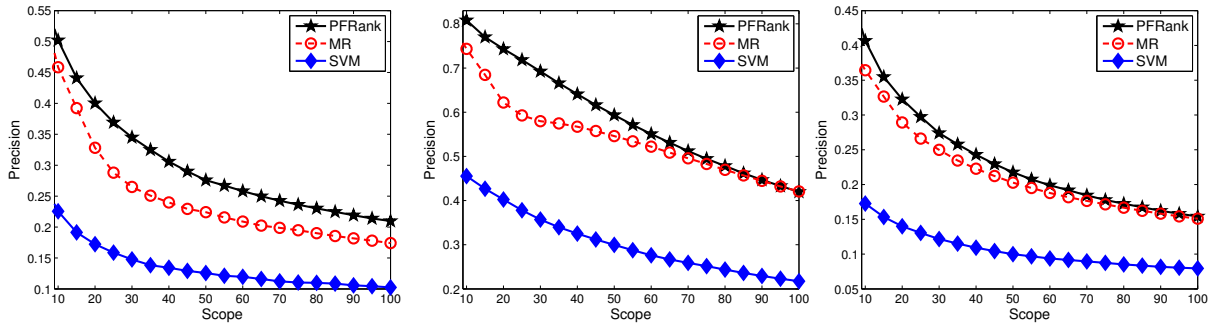
In general, it is appropriate to present 20 images on a screen. Putting more images on a screen might affect the quality of the presented images. Therefore, precision@20 is especially important. Figure 5.5, Figure 5.6 and Figure 5.7 show the precision@20 scores of PFRank with respect to different values of λ_1, λ_2 and λ_3, respectively. Note that within each data set, here we only randomly choose one image from each category as queries and present the averaged performance scores just to speed up the experiments. As can be seen, our PFRank algorithm is generally not very sensitive to the parameters (especially on the Yale-B data set), and outperforms the other two methods over a wide range of parameters.

5.6.3 Document Retrieval

In this experiment, we apply different ranking algorithms to the task of document retrieval, which is also studied in [90]. Given a document as a query, we want to rank the documents in our database according to their relevance to the query. Here we totally use three document data sets, all of which have category labels representing different topics. Then documents sharing the same category label (or topic) as the query document are judged relevant. Similar as before, for our PFRank algorithm, we perform cross-validation and set the dimensionality of the manifold (d) to be 2 for all the three document data sets. All the other experimental settings are the same as the previous image retrieval experiment. Here we provide a description of the data corpora we use.

The first corpus we use is the popular 20 Newsgroups[2] data set. This corpus is collected and originally used for document classification by Lang [44]. We use the "bydate" version which contains 18846 documents, evenly distributed across 20 categories. This corpus contains 26214

[2] http://people.csail.mit.edu/jrennie/20Newsgroups/



(a) 20 Newsgroups (b) TDT2 (c) Reuters

Figure 5.8: The average precision-scope curves of different algorithms on three document corpora.

distinct terms after stemming and stop word removal. Therefore, each document is represented by a 26214-dimensional term-frequency vector.

The second corpus is the TDT2 corpus[3], which consists of data collected during the first half of 1998 and taken from six sources, including two newswires (APW, NYT), two radio programs (VOA, PRI) and two television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In this corpus, we also removed those documents appearing in two or more categories and use the largest 30 categories thus leaving us with 9394 documents, each of which is represented by a 36771-dimensional term-frequency vector.

The third corpus is the Reuters-21578 corpus[4]. This corpus contains 21578 documents in 135 categories. In our experiments, we discard those documents with multiple category labels, and select the largest 30 categories. It left us with 8067 documents, each of which is represented by a 18933-dimensional term-frequency vector.

Performance evaluation

We show the average precision-scope curves generated by different ranking algorithms in the document retrieval task in Figure 5.8. Similar as before, our proposed PFRank algorithm performs the best on all the three data sets. In the document retrieval task, both PFRank and MR perform much better than SVM, indicating that considering the manifold structure in the documents could be even more useful than in the images. As the scope (n) becomes larger, the precision scores of

[3] http://www.nist.gov/speech/tests/tdt/tdt98/index.html
[4] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 5.4: Performance evaluated by different metrics on the 20 Newsgroups data set

	NDCG@10	NDCG @20	Recall@10	Recall@20	Recall@50	MAP
SVM	0.271	0.217	0.063	0.090	0.148	0.163
MR	0.519	0.406	0.109	0.148	0.236	0.263
PFRank	**0.550**	**0.462**	**0.119**	**0.181**	**0.288**	**0.312**

Table 5.5: Performance evaluated by different metrics on the TDT2 data set

	NDCG@10	NDCG @20	Recall@10	Recall@20	Recall@50	MAP
SVM	0.479	0.434	0.132	0.208	0.327	0.384
MR	0.777	0.681	0.199	0.292	0.586	0.668
PFRank	**0.832**	**0.778**	**0.260**	**0.409**	**0.643**	**0.761**

Table 5.6: Performance evaluated by different metrics on the Reuters data set

	NDCG@10	NDCG @20	Recall@10	Recall@20	Recall@50	MAP
SVM	0.196	0.165	0.160	0.201	0.263	0.215
MR	0.391	0.329	0.283	0.381	0.532	0.395
PFRank	**0.452**	**0.377**	**0.427**	**0.534**	**0.657**	**0.515**

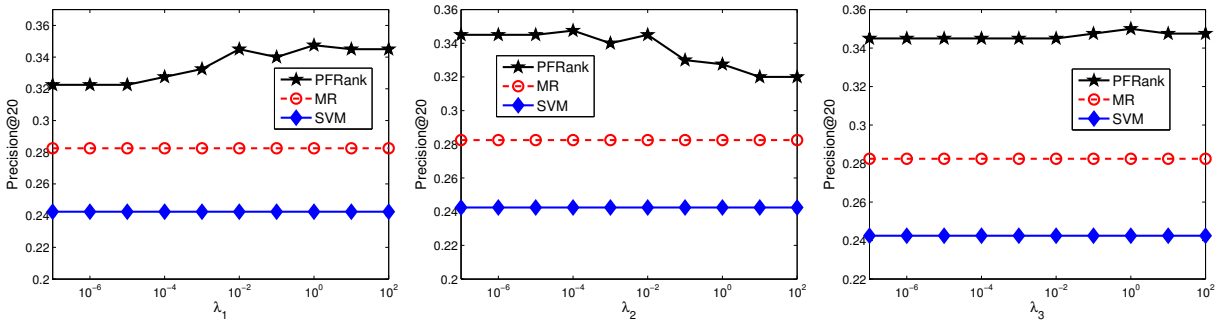
PFRank and MR generally become similar.

We also present the NDCG, recall and MAP scores of various algorithms on the three corpora in Table 5.4, Table 5.5 and Table 5.6, so as to give a comprehensive view of the document retrieval performance. As can be seen, our proposed PFRank algorithm consistently outperforms the other two methods in all the evaluation metrics, indicating very reliable performance. MR performs the second best, which is consistent with the precision-scope curves.

Model Selection

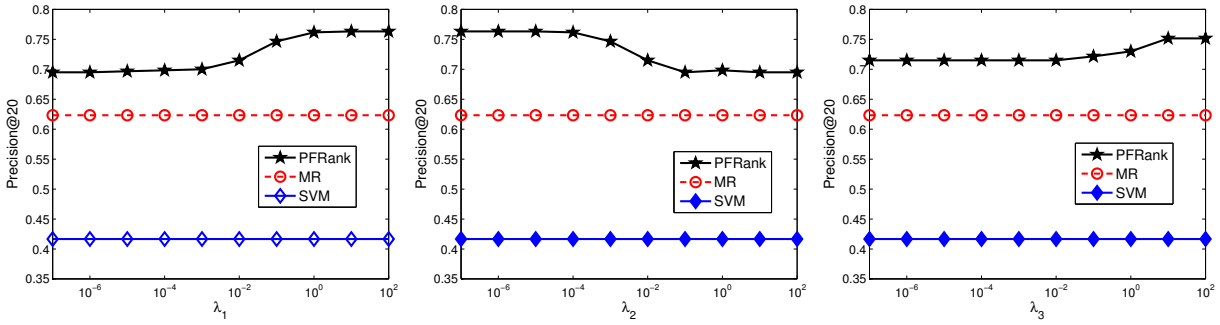
Similar to the image retrieval experiments, here we also study the impact of the parameters on the performance of our proposed PFRank algorithm in the document retrieval task. We fix all the other parameters the same as before, and let one of $\{\lambda_1, \lambda_2, \lambda_3\}$ vary. We show the precision@20 scores of PFRank with respect to different values of λ_1, λ_2 and λ_3 in Figure 5.9, Figure 5.10 and Figure 5.11 for the three document corpora, respectively. Within each document corpus, we still only randomly choose one document from each category (topic) as queries and average the performance scores in order to speed up the experiments.

In the document retrieval task, we can also see that our proposed PFRank algorithm is generally not very sensitive to the parameters. Over a wide range of parameters, PFRank can outperform the other two algorithms.



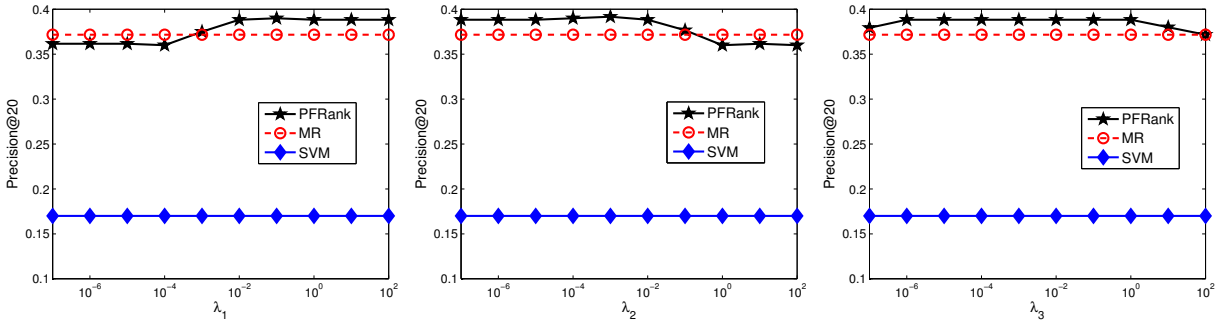
(a) Varying λ_1 (b) Varying λ_2 (c) Varying λ_3

Figure 5.9: Model selection on the 20 Newsgroups data set.



(a) Varying λ_1 (b) Varying λ_2 (c) Varying λ_3

Figure 5.10: Model selection on the TDT2 data set.



(a) Varying λ_1 (b) Varying λ_2 (c) Varying λ_3

Figure 5.11: Model selection on the Reuters data set.

Chapter 6

Relevance Search on Heterogeneous Graphs: A Meta Path Based Approach

6.1 Overview

Discovering strong relevance relationships between heterogeneous entities is a fundamental problem. By strong relevance we mean the relevance supported by rich relevance contexts in the data. Given an entity (e.g., a disease) as a query, a user may be interested in browsing other entities of heterogeneous types (e.g., drugs) that have strong relevance relationships with the queried entity. With the discovery of strong semantic relationships between entities, huge knowledge graphs can be built, and the user can navigate from one entity to other related entities and quickly find the information he/she is searching for.

Data may come from many different sources, with the graph data being one important form as described above. In this study, we try to perform relevance search among heterogeneous biomedical entities, and we start with unstructured biomedical data for a couple of reasons: 1) For structured data such as graphs, it may not be so challenging because basic relationships, such as customers "purchase" items, are explicit already, which could be used to derive strong relevance [34, 35, 74, 70, 46]. 2) It is much more challenging to find such relationships in text data, which is unstructured, noisy, and entity relationships are deeply hidden. Moreover, text data are ubiquitous, in huge amount, and being updated constantly. Mining entity relationships from text data is thus not only imperative, but also will greatly extend the scope of our study and improve the applicability of our proposed graph-based methods. 3) The rich biomedical domain knowledge offers the feasibility of extracting entities from unstructured biomedical data. However, entity recognition from many other domains is still an unsolved problem.

In the biomedical domain, drug discovery studies [65, 25, 16, 63] can only detect drugs that are known to treat certain diseases, and cannot discover strong relevance between drugs and diseases

that are not explicitly written in the text. Recommendation systems suggest the items that the users are likely to be interested in [67, 83, 23]. However, the data there are usually structured and the systems require the availability of training data (*e.g.*, some users are interested in certain items). Recent studies on similarity search in heterogeneous graphs, such as PathSim [74], explore a meta path based similarity measure. Nevertheless, their similarity measure is defined for comparing nodes of the same types (*e.g.*, similarity between authors in a bibliographic network). [70] first proposed to study the relevance between heterogeneous entities. However, their similarity measure is based on pairwise random walk which may not be able to capture the subtlety of the path-constrained strong relevance relationships as indicated in our experiments.

Based on these considerations, we propose our approach, which contributes to the state-of-the-art in the following aspects: (1) the method constructs a biomedical entity correlation graph from unstructured data, extending the scope of our previous graph-based study to unstructured text data; (2) the method extends the meta path based relationship analysis [74] to heterogeneous types of biomedical entities and infers top-k most effective meta paths from data given two types of entities that we aim to mine strong relevance between; (3) our new approach, EntityRel, proposes a new measure for computing the context-aware relevance between two heterogeneous entities; and (4) our experiments and performance comparison with several existing methods demonstrate the effectiveness of our method, with interesting results for the strong relevance discovery between drugs and diseases.

The biomedical entity correlation graph maintains basic entity relationships that can be straightforwardly found in unstructured text data, i.e., weighted co-occurrence. Based on it, the collection of paths linking two heterogeneous entities e_i and e_j offer rich semantic contexts for their relationships. However, not all paths carry the same semantics. For example, "tretinoin – skin – acne" indicates a therapeutic relationship between drug "tretinoin" and disease "acne", while "Vitamin A – toxicity – acne" indicates a side-effect relationship. Therefore, the relevance type depends on the contexts in paths. Our proposed approach, EntityRel, is such a context-aware relevance measure. Without loss of generality, we predefine 5 types of biological entities for constructing the entity correlation graph, which are: Drug, Compound, Disease, Target and MeSH. Based on them, we can define path types like "Drug – Target – Disease" or "Drug – MeSH – Disease", named as

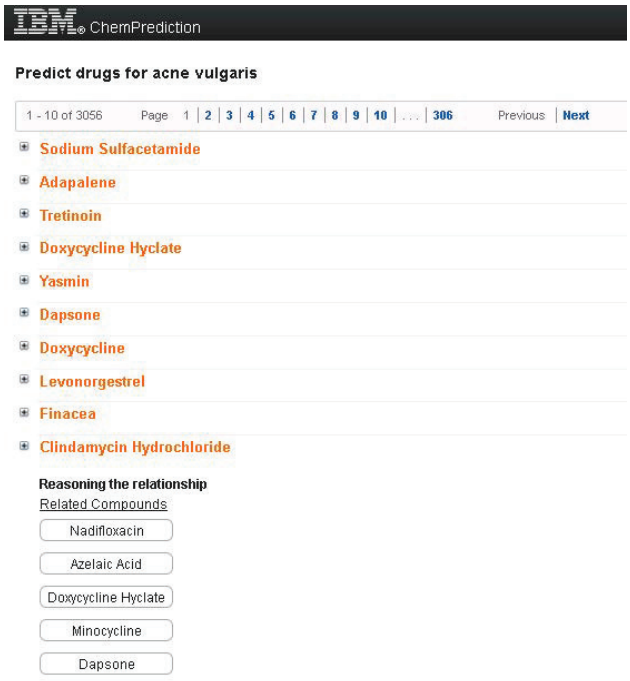


Figure 6.1: Drug search engine demo.

meta paths in the work. For example, "tretinoin – skin – acne" is one path instance of meta path "Drug – Target – Disease". In our approach, EntityRel, we assume that the relevance is only meaningful under path contexts constrained by certain meta path. For example, if we use all paths following the pattern "Drug – Target – Disease" as contexts, the discovered relationships between drugs and diseases are very likely therapeutic relationships. More specifically, we name the set of entities (excluding e_i and e_j) in these paths as "reasoning entities", which are used to reason the discovered relevance relationships.

Consequently, one natural question is: what kinds of paths should we use for mining the strong relevance between heterogeneous entities? The definition of "strong" relevance is a data dependent concept: some types of relevance might be strong and some types might be weak, depending on how rich the corresponding relevance contexts provided by the data can be. In this work, given two types of entities, we automatically pick up top-k meta paths from the data, such that the relevance contexts defined by these meta paths in data are relatively richer than other types of contexts. Based on these rich contexts, we are supposed to discover so-called "strong" relevance between the given two types of entities.

In addition to the theoretical contribution, we also implement a prototype drug search engine inside IBM. Figure 6.1 shows a real example in our demo system, where a user submits a disease "acne"[1] and searches for strongly relevant drugs. All the top 10 returned results are FDA-approved drugs for treating acne. Specifically, the 10-th drug "Clindamycin Hydrochloride" only co-occurs with "acne" and its synonyms 5 times in more than 20 million MEDLINE articles, which cannot be discovered by simple co-occurrence methods easily. Note that the correctness of strong relevance depends on the reasoning entities of the discovered relationship. All the five reasoning compounds (Nadifloxacin, Azelaic Acid, Doxycycline Hyclate, Minocycline, Dapsone) in the paths that contribute most to this discovery result clearly indicate that the strong relevance found by us between "Clindamycin Hydrochloride" and "acne" is a valid therapeutic relationship. On the contrary, if we use similar contexts to reason the relationship of "Vitamin A" (co-occur with acne 22 times) or "Insulin" (co-occur with acne 21 times) with "acne" , the relationship will be wrong despite that these two drugs are relevant to disease "acne" in other ways. For example, to treat acne, large doses of Vitamin A must be given, which then results in Vitamin A toxicity; acne has an effect of insulin resistant. These relationships have to be detected by other correlation contexts, such as "Symptom"-typed entities. In this work, when we judge the correctness of the discovered strong relevance, we utilize the set of reasoning entities involved in the relevance discovery.

6.2 Problem and Framework

Given an unstructured biomedical text data corpus $\mathcal{D}$ and K types of predefined biological entities $E_1, \ldots, E_K$, our problem is to automatically discover the strong relevance relationships between any pair of entities e_i and e_j strongly supported by $\mathcal{D}$, where e_i and e_j can belong to either the same entity type or different entity types. As a more general case, in this work we focus on the relevance relationships across heterogeneous entity types. We annotate $E(e_i)$ as the entity type name of e_i and $|E(e_i)|$ as the number of entities of type $E(e_i)$.

Formally, we quantify the relevance relationship between two heterogeneous entities e_i and e_j as a relevance score $R(e_i, e_j)$. The computation of $R(e_i, e_j)$ depends on the observed correlations of e_i and e_j in data $\mathcal{D}$. Possibly the simplest correlation between e_i and e_j is the number of co-occurrence

[1]The hit disease "acne vulgaris" is its synonym.

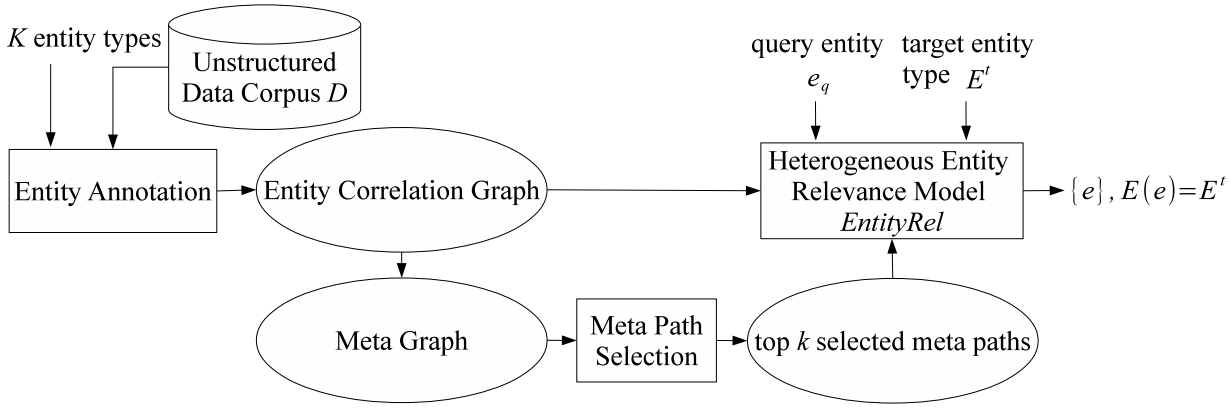


Figure 6.2: System framework of EntityRel.

in $\mathcal{D}$. However, the simple co-occurrence model can not effectively capture the correlation contexts of e_i and e_j. For example, given a frequent sentence "Which is able to treat acne, doxycycline or tetracycline?" in MEDLINE, it is hard to tell the drug entity "tetracycline" is relevant to the disease entity "acne" or not.

We observe that, the correlation contexts between two entities can be manifested by other entities that frequently co-occur with both. For example, given the frequent sentence "Acne is a disease that affects the skin's oil glands." in MEDLINE, we know that to treat the disease "acne", the organism entity "skin" is one kind of targets. Then, given another frequent sentence "Tetracyclines are oral antibiotics often used to treat skin diseases." in MEDLINE, we know that the function scope of drug "tetracycline" covers the target entity "skin". Thus, the target entity "skin" effectively links the drug entity "tetracycline" and the disease entity "acne" together and implies their relevance.

The rich context information in unstructured data corpus $\mathcal{D}$ can be represented by an undirected Entity Correlation Graph $\mathcal{G}$. In graph $\mathcal{G}$, the nodes are heterogeneous entities and the edge between two entity nodes represents the fact that these two entities once co-occur in data $\mathcal{D}$. Given one node e_i, its neighborhood set $N(e_i)$ includes all other entities that co-occur with it in data. Given two example entities "tetracycline" and "acne", we can extract a number of paths linking them, e.g., "tetracycline – skin – acne", "tetracycline – protein synthesis inhibitor – bacterial infection – acne" etc., from the graph $\mathcal{G}$. All these paths collectively serve as the correlation contexts for "tetracycline" and "acne".

Compared to the unstructured data corpus $\mathcal{D}$, the Entity Correlation Graph $\mathcal{G}$ is structured and easy to analyze and store. Representing the unstructured text as such a graph enables the task of entity relationship discovery to be formulated as searching relevant heterogeneous entities by traveling the graph. For example, for discovering the relationships between drugs and diseases, the user inputs a disease entity "acne" and then search all drug entities reachable in the graph. Without the loss of generality, we formulate $R(e_i, e_j)$ as a search problem: computing $R(e_q, e)$ by treating $e_q = e_i$ as the query entity, $e = e_j$ as the searching target entity, and $E(e) = E(e_j)$ which is the target entity type. The whole framework is given by Figure 6.2.

6.3 Correlation Graph Construction

Many existing work in graph-based entity search has already assumed the existence of the entity graph [34, 35, 74, 70, 46]. However, in this work, how to automatically generate an Entity Correlation Graph $\mathcal{G}$ from the unstructured data corpus $\mathcal{D}$ remains challenging, which is discussed in this section.

6.3.1 The unstructured data corpus $\mathcal{D}$

We use MEDLINE [2], a bibliographic database of life sciences and biomedical information, as the knowledge base to discover entity relationships in this work. The abstracts of all 20,642,063 biomedical documents to date consist of the unstructured data corpus $\mathcal{D}$.

We select 5 types of biological entities, *Drug*, *Disease*, *Compounds*, *Target* and *MeSH terms*, to study in this work. In total, we predefined 5,867 FDA-approved drugs[3]; a dictionary of 4,244 diseases extracted from human disease ontology[4]; a set of 2,254 small-molecule chemical compounds with explicit drug indications from the Chemical Entities of Biological Interest (ChEBI) database[5]; a dictionary of 11,280 targets made up of 4 sub types: tissue, cell-line, protein, organism; and a set of all 17,347 leaf MeSH terms in the MeSH tree[6], which are used as the meta-data to index

[2] http://www.nlm.nih.gov/bsd/pmresources.html
[3] http://www.accessdata.fda.gov/scripts/cder/drugsatfda/
[4] http://www.obofoundry.org/cgi-bin/detail.cgi?id=disease_ontology
[5] http://www.ebi.ac.uk/chebi/. Note that drugs belong to compounds. In this work, we treat them differently as they originate from different sources orthogonally.
[6] http://www.nlm.nih.gov/mesh/

medical articles in MEDLINE by NIH. All the above entities consist of the node set in the Entity Correlation Graph G.

6.3.2 Entity annotation in text

Given the MEDLINE corpus and 5 types of biological entities, the first step is to annotate those entities in the MEDLINE articles.

For Disease and Target annotators, we adopted a dictionary-based method to loop up entities in the text based on the exact string match.

For Drug annotator, considering a drug usually contains a number of synonyms like brand name etc., our method is dictionary-based and enhanced by synonyms extracted from CHEMBL[7].

For Compound annotator, we designed a context-aware Conditional Random Field model [82], where both compound structural features and text content features are used to infer the labeling of text. To reduce the ambiguity of compound names, we converted all compound substances to their International Chemical textual identifiers (InChI) first, and then used the InChIKey, a fixed length (25 character) condensed digital representation of the InChI, to represent each compound.

For MeSH annotator, as all articles in MEDLINE already have $10 \sim 15$ MeSH terms labeled by human, we simply used these labeled MeSH terms as the annotation results.

6.3.3 Correlation weight in correlation graph

After entities are annotated in text, we can easily add an edge between two entities e_i and e_j in the Entity Correlation Graph if they are annotated in the same set of articles. The left question is to find a reasonable weighting function w_{ij} for the edge. Straightforwardly, we can simply use the raw number of co-occurrence to weigh the edge $w_{ij} = co(e_i, e_j)$, where $co(e_i, e_j)$ is the number of articles where both e_i and e_j are annotated in the text. However, this simple method largely favors those popular entities appearing in many articles. Instead, we propose to compute w_{ij} with full consideration of both the relevant frequency that two entities co-occur and the popularity of each entity. Following the classical TF-IDF model in information retrieval, we assume a large w_{ij} implies:

[7] https://www.ebi.ac.uk/chembl/

(1) e_i and e_j co-occur frequently;

(2) e_i (or e_j) occurs rarely with other entities of the type $E(e_j)$ (or $E(e_i)$).

To satisfy the first condition, we design a normalized symmetric frequency function as

$$freq(e_i, e_j) = \frac{co(e_i, e_j)}{\left(\sum_{e_y \in E(e_j)} co(e_i, e_y) + \sum_{e_x \in E(e_i)} co(e_x, e_j)\right)/2}.$$

To satisfy the second condition, we define $ief(e_i, e_j)$ which represents the "inverse entity frequency" to measure whether entities e_i and e_j are common or rare within all the co-occurrence between entities of type $E(e_i)$ and $E(e_j)$:

$$ief(e_i, e_j) = \log \frac{(|E(e_i)| + |E(e_j)|)/2}{1 + \big(|N(e_i) \wedge E(e_j)| + |N(e_j) \wedge E(e_i)|\big)/2}, \tag{6.1}$$

where $N(e_i) \wedge E(e_j)$ represents the joint set of entities who are neighborhoods of e_i and have the same entity type as e_j.

Finally, we have the symmetric correlation weighting function

$$w_{ij} = freq(e_i, e_j) \times ief(e_i, e_j). \tag{6.2}$$

Note that this correlation weighting function is different to our target function $R(e_i, e_j)$. The former is designed to weigh the correlation between two entities without considering the global correlation graph structure and other type of entities. It can be treated as one kind of local affinity measure between two entities. This function can be used as one naive solution of $R(e_i, e_j)$. But, this naive solution undoubtedly has many limitations. First, the correlation contexts which have been previously shown to be effective in linking entities are lost. Second, it cannot find those entities who never directly co-occur with the query entity.

In this work, we use w_{ij} as the elementary edge weighting function for the Entity Correlation Graph G; and then explore other more sophisticated graph travel methods for the entity relationship discovery based on the graph.

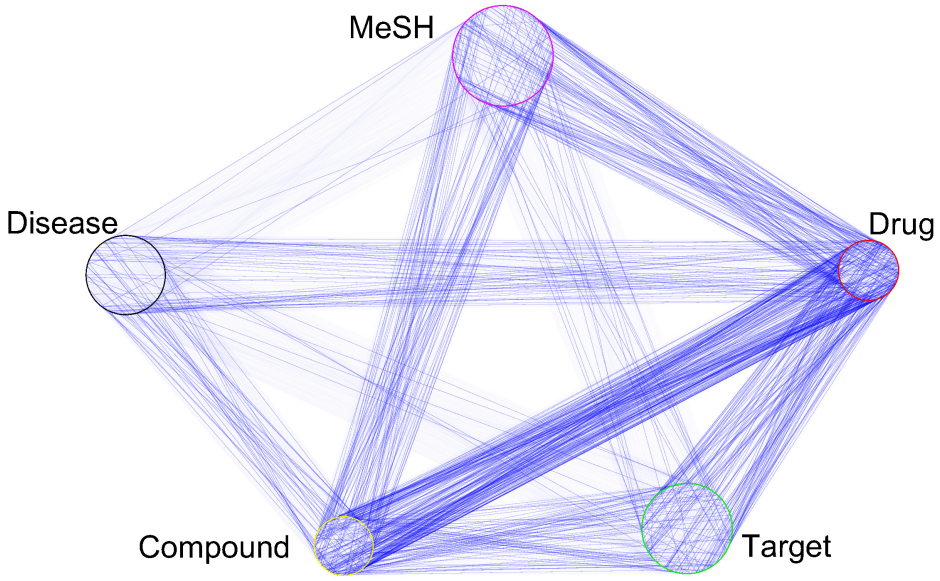


Figure 6.3: Entity Correlation Graph $\mathcal{G}$. One edge = 1,000 links in data. The size of circle is proportional to # of entities.

6.3.4 Properties of entity correlation graph

The constructed Entity Correlation Graph visualized by Figure 6.3 has many interesting properties. It tells some meta paths (*i.e.* Drug – Compound) are much denser than others (*i.e.* MeSH – Disease). To uncover strong relevance from the data, we may only focus on those dense meta paths which are strongly supported by data.

The degree distributions of $\mathcal{G}$ and individual entity type are depicted in Figure 6.4; One interesting finding is: various entity types have various degree distributions, resulting in various graph structures. For example, both Disease and Compound have very flat power law slope, indicating that their node degrees are more uniformly distributed. Relatively, the other entity types contain fewer highly connected nodes. This finding discloses that, if we treat the entire graph as a homogeneous graph without differentiating entity types and then randomly surfer in graph, some entity types will be favored and some entity types are not reachable. Therefore, traditional methods to compute entity relevance in a homogeneous graph like SimRank [34] and PathSim [74] are not suitable for relevance between heterogeneous entities.

Graph $\mathcal{G}$ is a typical "small world". 91.75% of its nodes belong to a giant connected component. The average distance between two nodes in this giant component is 2.0663, indicating that starting

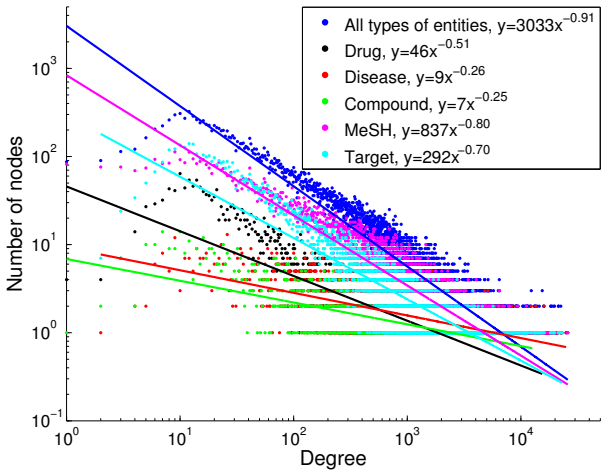


Figure 6.4: Degree distribution of the nodes in $\mathcal{G}$.

from one node, we can quickly arrive at other nodes. The "small world" phenomenon in $\mathcal{G}$ offers rich contexts (numerous different paths) between two nodes.

6.4 Meta Path for Correlation Contexts

With entity correlation graph, the next step is to learn strong correlation contexts from it for discovering strong entity relevance relationships in this section.

6.4.1 Strong meta paths as contexts

As we mention before, given the correlation graph $\mathcal{G}$, we can formulate the task of entity relevance relationship discovery as searching relevant heterogeneous entities in the graph. For example, given the disease "acne", what are the similar drugs in the graph? The objective of the problem is to infer the relevance score denoted by $R(e_q, e)$, given the query entity e_q and one entity e with the target entity type.

In Section 6.3.4, the graph $\mathcal{G}$ has been shown to be extremely complicated and overwhelmed across different entity types. Given two heterogeneous entities e_q and e, there exists numerous number of paths linking them if we do not constrain the length of path. More specifically, due to the "small world phenomena" in $\mathcal{G}$, most pairs of entities can be linked together within 2 steps. Apparently, we do not want to recommend all entities to the query. From the complex graph $\mathcal{G}$,

we observe that,

(1) If a path is too long, it may cause concept drift and link two irrelevant entities together. For example, a path "acne (Disease) – skin (Target) – muscle (Target) – Ryanodine Receptor Calcium Release Channel (MeSH) – rycals (Drug)" links the drug "rycals" that treats skeletal muscle to an irrelevant disease "acne".

(2) To study the strong relevance relationships between two types of entities, some types of paths are preferred to others for a given data. For example, in Figure 6.3, the meta path "Drug – Compound – Disease" has much more path instances than the meta path "Drug – MeSH – Disease". The former thus implies stronger relevance than the latter in data. Or, the relevance implied by the latter is less confident, due to the sparsity of the data.

Our observations indicate that, to compute $R(e_q, e)$, a subset of relatively short paths between e_q and e should be extracted out from the graph $\mathcal{G}$ to serve as the correlation contexts. Finding a maximum induced connected subgraph (all paths are connected to e_q and e) with certain property (i.e., following the above two observations) from $\mathcal{G}$ is a classic NP-complete problem. For that reason, we design an approximated solution for context selection in the following.

We first convert the context selection problem into a meta path selection problem. As long as a meta path is selected, all its path instances will be selected as contexts. The meta path is formally defined as follows.

Definition 5. **Meta Path.** A meta path m of length l is a sequence of nodes in the form of $E_{x_1} \overset{A_{x_1,x_2}}{\longrightarrow} E_{x_2} \overset{A_{x_2,x_3}}{\longrightarrow} \cdots \overset{A_{x_l,x_{l+1}}}{\longrightarrow} E_{x_{l+1}}$ where $x_y \in [1, K]$, $y \in [1, l]$. $A_{x_y,x_{y+1}}$ defines a composite correlation weight between two entity types E_{x_y} and $E_{x_{y+1}}$; and $A_{x_1,x_2} \cdots A_{x_l,x_{l+1}}$ defines a composite correlation weight W_m for the path m:

$$\begin{aligned} W_m = A_{x_1,x_2} \cdots A_{x_l,x_{l+1}} &= \prod_{y=1}^{l} A_{x_y,x_{y+1}} \\ = \prod_{y=1}^{l} \frac{\sum_{e_i \in E_{x_y}, e_j \in E_{x_{y+1}}} w_{ij}}{|E_{x_y}| \times |E_{x_{y+1}}|}, & \end{aligned} \tag{6.3}$$

where w_{ij} denotes the correlation weight of the edges belonging to the paths (rf. Eq. 6.2).

Meta path defines the sequence pattern of regular paths. A meta path with large path weight

implies that the regular path instances following its pattern have a large correlation weight on average. Our meta path weight definition favors the short paths. To compute $R(e_q, e)$ by ranking all meta paths starting from $E(e_q)$ and ending at $E(e)$ w.r.t their weights, if we only choose top-k meta paths, the discovered relevance relationships are relatively strong in data.

One may wonder that why our context selection strategy is at the meta path level, not at the path instance level. The reasons have two folds. First, context selection at the path instance level has to be computed on-line for each query. Due to the large size of entity correlation graph, it is thus not efficient to deploy in the real time. Instead, context selection at the meta path level can be computed off-line. Second, no single data can cover all possible entity relationships in the real world. Thus, the path instance level context selection may overfit the data. Alternatively, context selection at the meta path level strikes a good balance between data sparsity and the average performance.

6.4.2 Meta graph for meta path selection

Given K types of entities, there are $K(K-1)/2$ different pairs of types. For each pair, we need pre-compute all possible meta paths and rank them w.r.t. their weights. One simple yet efficient way to enumerate the meta paths is to maintain a meta graph in memory, which is defined as follows.

Definition 6. **Meta Graph.** Given the entity correlation graph $\mathcal{G}$ and K types of entities $E_1, \ldots, E_K$, a graph $\mathcal{G}^m$ is called a meta graph over $\mathcal{G}$ when its nodes are one of K entity types and the weight between two entity types E_i and E_j is defined as $A_{i,j}$ that follows the composite correlation weight definition in Eq. 6.3.

The meta graph $\mathcal{G}^m$ actually defines a $K \times K$ pair-wise weight matrix for K entity types. It is a dense graph because $A_{i,j} > 0$ as long as there exists one entity of type E_i co-occurs with another entity of type E_j in data. It is a symmetric graph as $A_{i,j} = A_{j,i}$. The diagonal elements in the matrix (when $i = j$) indicate the self-correlations of one entity type, which cannot be ignored because it is common that entities of the same type co-occur with each other. The meta graph can be seen as a summary of the original large entity correlation graph at the entity type level. For example, based on our entity correlation graph $\mathcal{G}$ built upon MEDLINE and five types of entities

Algorithm 1: Top-k meta paths selection

Input: Meta graph $\mathcal{G}^m$ and two question entity types E_i and E_j for entity relationship discovery; k
Output: Top-k meta paths in terms of path weights.
Initialize two empty sets A and $\mathcal{O}$;
repeat
  Find the meta path m not in $\mathcal{O}$ with the highest path weight from $\mathcal{G}^m$. Path m must have the form of $E_i \xrightarrow{A_{i,x_2}} E_{x_2} \xrightarrow{A_{x_2,x_3}} \cdots \xrightarrow{A_{x_l,j}} E_j$;
  Insert all pairs of composite correlations $A_{x_y,x_{y+1}}$ in path m into the set A;
  if $|A|$ *increases* **then**
    Insert path m into $\mathcal{O}$;
  end
until $|\mathcal{O}| = k$;
return $\mathcal{O}$

(Disease, Drug, Compound, Target, MeSH), we build a meta graph $\mathcal{G}^m$.

Based on the meta graph and the starting/ending entity types, we can efficiently enumerate all possible meta paths. Recall that we have two principles to select meta paths: 1) the length is not too long; 2) the path has high weight. As our meta path weighting function (Eq. 6.3) has implicitly favored short paths already, a simple greedy algorithm that travels all meta paths from the highest path weight to the lowest path weight is sufficient to our goal, as shown in Algorithm 1. Each time a new meta path m is selected, it must contain at least one new type of composite correlation, or a new pair of entity types in another word. This heuristic prevents the information self loop in meta path and thus largely limits the scope of path candidates.

Example results of top-k meta paths selection

By setting $k = 5$ and selecting two entity types Drug and Disease as an example, the top meta paths generated by Algorithm 1 from our data are listed as follows:

- Drug - Disease,
- Drug - Drug - Disease,
- Drug - Compound - Disease,
- Drug - Disease - Disease,
- Drug - MeSH - Disease.

These five meta paths collectively generalize the correlation contexts for any $\langle drug, disease \rangle$ pair while measuring their strong relevance.

6.5 Meta Path Based Heterogeneous Entity Relevance Model

For the problem of searching relevant heterogeneous entity e of type E^t in graph $\mathcal{G}$ for a query entity e_q, the previous section selects top-k meta paths as the relevance contexts. The top-k meta paths collectively define a subgraph $\mathcal{G}' \in \mathcal{G}$. Based on it, our core task is to compute $R(e_q, e)$.

6.5.1 Review related work in computing $R(e_q, e)$

The related work in computing $R(e_q, e)$ can be categorized along two dimensions: context-aware and context-agnostic; homogeneous and heterogeneous.

Personalized PageRank [35] computes the probability of a random walker starting from e_q can arrive at e in the graph as $R(e_q, e)$, where the teleport only switches to the query entity e_q. As a general-purpose graph similarity measure, Personalized PageRank is a context-agnostic model designed for a homogeneous graph. Its variation, called Path Constrained Random Walk [46], is extended for heterogeneous graphs. It computes the probability of a random walker starting from e_q can arrive at e through constrained paths in the graph as $R(e_q, e)$. It is designed for a single meta path. These random walk models favor the popular entities undesirably and ignore the differences of various contexts inherited from various meta paths.

SimRank [34] is another context-agnostic model designed for the homogeneous graph. It iteratively computes $R(e_q, e)$ as the sum of similarities between their neighbors in the graph. The entity types of their neighbors are ignored. HeteSim [70] extended SimRank to the heterogeneous graph. Given a meta path, it computes the average fraction of information that can diffuse from the middle node of the path to two ends as $R(e_q, e)$. However, HeteSim only depends on the raw counts of paths without fully utilizing the rich contexts of these paths.

6.5.2 Context-aware relevance model

Our goal here is to design a novel relevance measure for two heterogeneous entities that fully considers the subtlety of different types among entities and the top selected meta paths as the

correlation contexts.

One straightforward way satisfying all the above conditions is a model "conditioned" on the top-k meta paths. Formally, we have

$$R(e_q, e) = \sum_m P(m) R(e_q, e, m) \tag{6.4}$$

In this way, $R(e_q, e)$ can be seen as a linear combination of the relevance over each meta path m. The meta path prior $P(m)$ can be learned in a supervised manner [45], which is however out of the scope of this work. In the work, we manually tune the weights of meta paths and put our focus on the computation of $R(e_q, e)$.

Based on the weight of the edges constituting the paths between two heterogeneous entities, we design a simple measure of $R(e_q, e, m)$:

$$R(e_q, e, m) = \sum_{e_q \rightsquigarrow e \in m} \left(\prod_{\langle e_i, e_j \rangle \in e_q \rightsquigarrow e} w_{ij} \right) \tag{6.5}$$

Following notations in [74], $e_q \rightsquigarrow e$ denotes a path instance (belonging to the meta path m). $\langle e_i, e_j \rangle$ denotes an edge belonging to path instance $e_q \rightsquigarrow e$, and w_{ij} is the weight of the edge. The above measure has the good symmetric property, i.e., $R(e_q, e, m) = R(e, e_q, m^{-1})$. Moreover, it fully takes into account the weight of the edges (and in turn the paths) instead of just doing simple path counting, therefore well utilizing the rich contexts encoded in the paths. This design is also consistent with the weight of meta paths defined in Eq. 6.3.

6.6 Experiments

In this section, we empirically evaluate the effectiveness of our proposed framework for estimating the relevance between heterogeneous entities. We begin with the experimental setup.

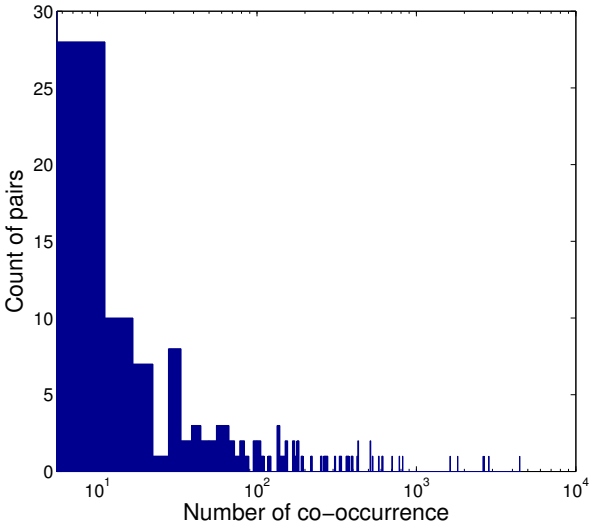


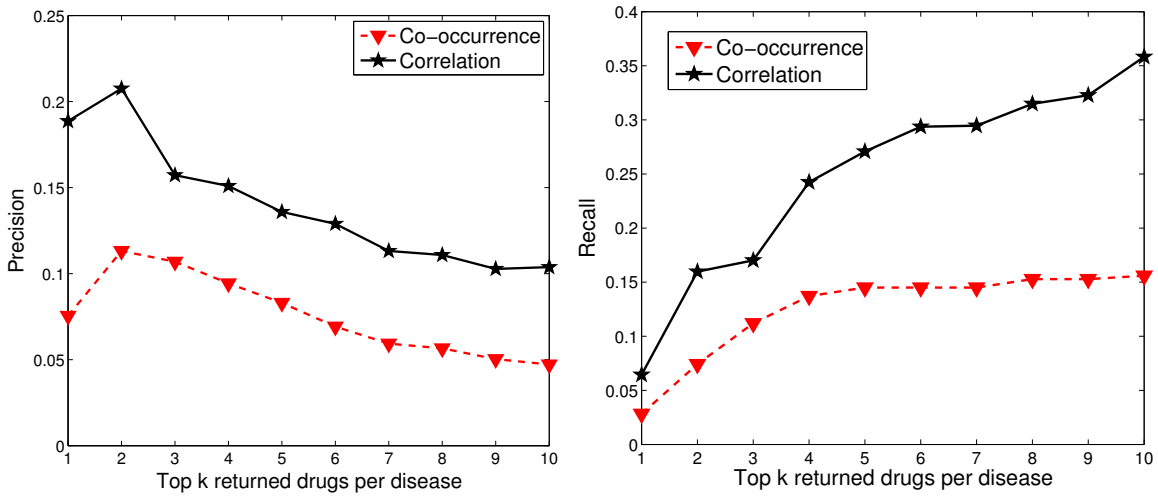
Figure 6.5: Histogram of # of times that the ground truth drug-disease pairs co-occur in text corpus $\mathcal{D}$.

6.6.1 Experimental setup

In order to evaluate the relevance estimation results generated by different algorithms, we sampled 199 unique drug-disease pairs from FDA's orange book[8] as the ground truth for the therapeutic relationships between drugs and diseases. We chose the therapeutic relationship as testing cases because it is one kind of strong relevance largely supported by the MEDLINE data. While sampling, we emphatically avoid those well-known drugs, as their relevance can be easily captured by their large amount of co-occurrences with diseases. The co-occurrence distribution of our ground truth drug-disease pairs is illustrated by Figure 6.5. It can be observed that most of the drugs that known to treat certain diseases co-occur rarely with the disease (typically no more than 10 times out of the 20 million abstracts). Therefore, the relevance relationship that we want to discover is really hidden in the text and can hardly be discovered by simply counting the raw co-occurrence numbers or natural language processing techniques.

Given a disease, all drugs in the database can be ranked according to the relevance scores, denoting how likely each drug is relevant to the disease. It is tricky to judge the "correct" returned drugs as we only have ground truths for the therapeutic relationship, not for strong relevance in

[8]http://www.accessdata.fda.gov/scripts/cder/ob/default.cfm. Among all the relevance relationships between different types of biological entities, we show the discovery results of the therapeutic relationships as an example since the results are easy to be evaluated by referring to FDA's orange book.



(a) Precision (b) Recall

Figure 6.6: Compare *correlation* to *co-occurrence*.

general. To evaluate the correctness of a returned drug, not only will the drug be compared with ground truths (for Recall), but also the reasoning entities will be manually checked by human experts to see if the inferred relationship falls in the treatment category (for Precision). We use the standard precision, recall and Mean Average Precision (MAP) [55] to evaluate the results for our problem. Precision is defined as the # drugs can treat the query disease based on human evaluation by the # of returned drugs. Recall is defined as the # drugs in ground truths divided by the # of returned drugs. Given a disease, let r_i be the judgement score of the drug ranked at position i, where $r_i = 1$ if the drug is known to treat the disease and $r_i = 0$ otherwise. Then we can compute the Average Precision (AP):

$$AP = \frac{\sum_i r_i \times \text{Precision@}i}{\#\text{ of drugs known to treat the disease}}$$

And MAP is the average of AP over all the diseases in our labeled set. We do not use Normalized Discount Cumulative Gain (NDCG) to measure the performance, since we can only manually judge whether a drug can or cannot treat the given disease, but do not have levels about how much one drug can treat one disease.

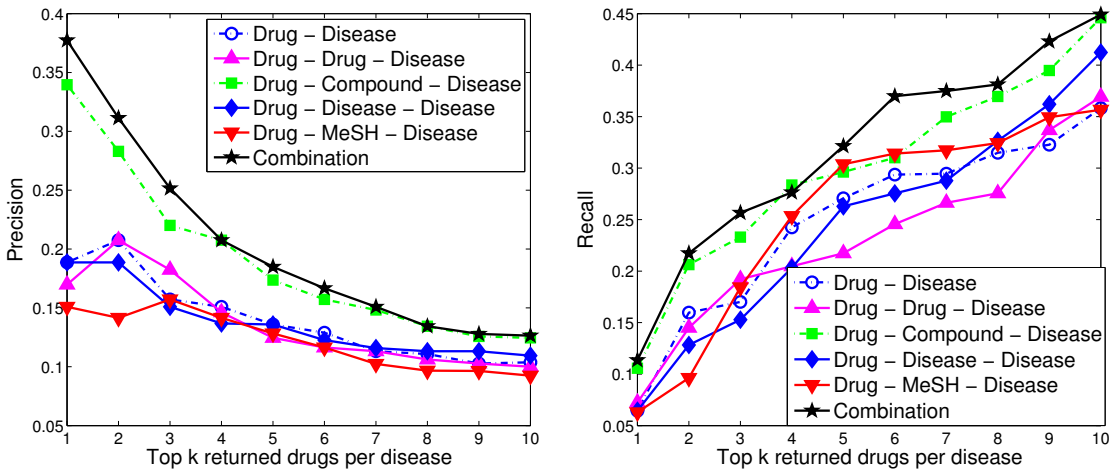
6.6.2 Correlation weight evaluation

We first evaluate the effectiveness of our proposed correlation weight function comparing to the raw co-occurrence count for entity correlation graph construction. As mentioned before, the correlation weight function could be used as one naive solution of $R(e_q, e)$, where e_q is the query disease and e is one drug. We show the average precision curves and recall curves of the co-occurrence based method (denoted by *co-occurrence*) and correlation weight function based method (denoted by *correlation*) with regarding to the top number of returned drugs per disease in Figure 6.6. We show the precision and recall measures for the top 10 returned drugs per disease since the top 10 returned results are the most important to the user and largely affect the user experience. The MAP scores for *correlation* and *co-occurrence* are 0.216 and 0.118, respectively, measuring the performance over the entire ranking list. The correlation weight based method performs much better than the raw co-occurrence based method on all the above evaluation metrics, meaning that our design of the correlation weight function in Eq. 6.2 is very reasonable to capture the direct correlation between two entities. So, we use Eq. 6.2 instead of the raw co-occurrence to construct the entity correlation graph.

6.6.3 Comparing different meta paths

We selected top 5 meta paths using Algorithm 1 (listed in Section 6.4.2). Based on a single meta path, we can find relevant drugs. We can also perform a weighted combination of the results generated by multiple meta paths. How to combine the results of different meta paths or how to set the weight for each path during combination is a difficult problem, which is left for future study. In this work, we manually tuned the weights and picked up the weights with the best performance.

Based on EntityRel, we compare the retrieval performance of individual meta paths and their combination. We show the average precision curves and recall curves of various meta paths and their combination in Figure 6.7, where the manually selected weights (similar to weights tuned by cross-validation) for the top 5 meta paths (shown in Section 6.4.2) are 0.01, 0.1, 0.78, 0.1, 0.01, respectively. Generally, we can see that combining the results generated by different meta paths performs equal or better than any single meta path, especially in the top few returned drugs. The MAP score comparison of different meta paths and their combination is shown in Table 6.1, where



(a) Precision (b) Recall

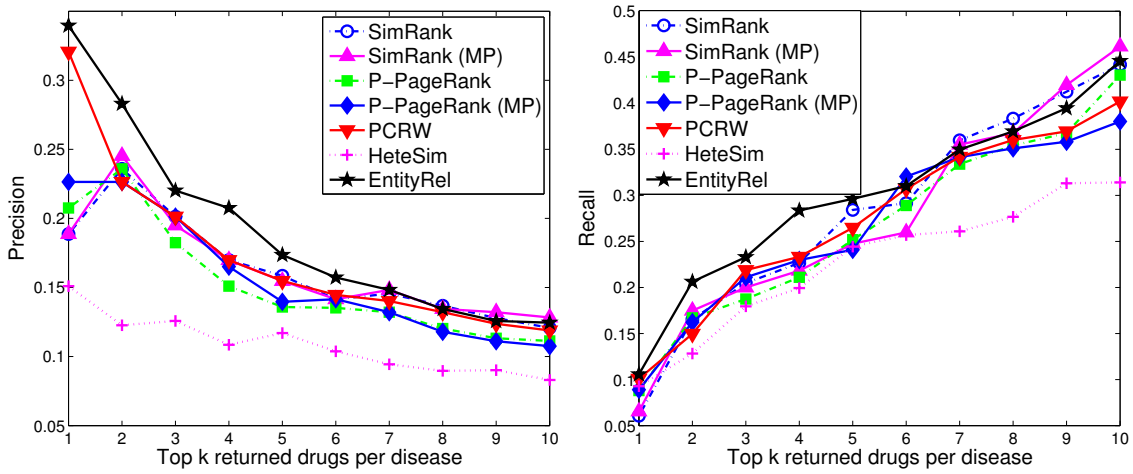
Figure 6.7: Compare different meta paths and their combination in Precision/Recall based on EntityRel.

Table 6.1: Compare different meta paths and their combination.

Meta Path	MAP
Drug - Disease	0.216
Drug - Drug - Disease	0.218
Drug - Compound - Disease	0.276
Drug - Disease - Disease	0.216
Drug - MeSH - Disease	0.203
Combination	**0.290**

we can see the combination method achieves the highest MAP score, indicating the best overall performance. Among the results generated by one single meta path, path "Drug - Compound - Disease" performs the best.

Remember the MAP score of the *correlation* method in the previous subsection is 0.216, which is much lower than both the combination method and the best result generated by one single meta path. This indicates that employing the top meta paths can generate better results than simply using the *direct* correlations between drugs and diseases as the relevance estimation, since the former encodes the interrelationships between multi-typed entities in a structured way.



(a) Precision (b) Recall

Figure 6.8: Compare EntityRel to related work.

Table 6.2: Compare EntityRel to related work in MAP.

Algorithm	MAP
SimRank	0.251
SimRank (MP)	0.254
P-PageRank	0.245
P-PageRank (MP)	0.244
PCRW	0.253
HeteSim	0.204
EntityRel	**0.276**

6.6.4 Comparing different methods on the same heterogeneous entity correlation graph

As mentioned before, the following state-of-the-arts can be used to estimate relevance between two homogeneous or heterogeneous entities. We adapted them on the same heterogeneous entity correlation graph $\mathcal{G}'$ generated by top 5 meta paths for a fair comparison.

- Personalized PageRank [35]. The damping factor is set as 0.9. By ignoring the type difference among entities and links, it can be run on two different graphs in our scenario: (1) the original correlation graph $\mathcal{G}$, named P-PageRank; and (2) the graph $\mathcal{G}'$ which only contains top 5 meta paths, denoted by P-PageRank (MP).

- *SimRank* [34]. Damping factor is set to 0.8. Has two versions similarly: SimRank on $\mathcal{G}$ and SimRank (MP) on $\mathcal{G}'$.

- *HeteSim* [70] run on its best meta path.
- Path Constrained Random Walk (*PCRW*) [46] run on its best meta path.

HeteSim, PCRW and our method EntityRelare all based on the best meta path selected from the top 5 meta paths. Although it is verified in the previous subsection that combining the results generated by multiple meta paths performs better than a single meta path, how to combine multiple meta paths without any label information is still an unsolved problem left for future study. Therefore, we simply run HeteSim, PCRW and our method on each of the top 5 meta paths and choose the best results for comparison.

It is worth noticing that both original SimRank and original HeteSim work on binary graphs only, considering whether two nodes are connected or not and ignoring the weight on the edges. We tried the original versions on the binary correlation graphs without using the weighted edges, and found that they performed rather poorly. Therefore, we show these two methods' results on the weighted correlation graph only.

From the average precision curves and recall curves shown in Figure 6.8, we can see that our EntityRel model leads the pack, especially when the number of returned drugs is small. PCRW performs the second best. Another observation is that SimRank performs similarly on the complete correlation graph $\mathcal{G}$ and the graph only containing selected meta paths $\mathcal{G}'$, and so does P-PageRank. This indicates that while reducing the time and space complexity largely, our top 5 selected paths capture most of the useful information in the original graph. The MAP scores of different algorithms are shown in Table 6.2. We can see our EntityRel is still the best, achieving 8.66% relative MAP score improvement over the second best algorithm. This indicates the reliable performance of EntityRel over the entire ranking list of returned drugs.

Chapter 7

Conclusions

In this thesis, two important and closely related research topics in mining networked data are presented: semi-supervised learning and relevance search.

For semi-supervised learning on homogenous graphs, I propose a novel label selection algorithm purely based on the graph structure, without label information and feature representation of the nodes [36]. One key advantage over existing methods is that the proposed method theoretically minimizes the expected prediction error of a popular graph-based semi-supervised learning model (and therefore improving its effectiveness) in a batch, offline mode. On heterogeneous networks, I investigate a new problem of using semi-supervised learning to classify nodes in a heterogeneous network, while simultaneously ranking the nodes according to their importance within each class, in order to provide informative class summaries [37]. A novel ranking-based classification algorithm called RankClass is proposed to iteratively solve this problem. During each iteration, the ranking distribution over the nodes of the network is calculated by authority propagation. The ranking results are then used to modify the network structure to allow the ranking model to improve the within-class ranking. Thus, the sub-network corresponding to each specific class is gradually extracted from the global network. The semi-supervised classification results generated in this way are very accurate and informative.

For relevance search on homogeneous graphs, a novel relevance function learning algorithm under the manifold assumption is developed from the vector field perspective [38]. Motivated by recent study about the relationship between vector fields and semi-supervised learning on the manifold, in this thesis, vector fields are employed to ensure the linearity of the relevance function with respect to the manifold, as well as to require the predicted relevance score of the query to be higher than that of its neighboring nodes. In this way, the relevance function learned by the proposed method decreases linearly, and therefore monotonically from the query node to other

nodes along the geodesics of the data manifold. Hence the ranking order along the geodesics is well preserved. For heterogeneous networks, the relevance search problem between different types of networked data is investigated. In order to make the method more widely applicable, a heterogeneous network is first built from the unstructured text data. Then a network-based relevance search model is designed based on the intuition that two nodes of heterogeneous types are strongly relevant if they are linked by many paths with high weight following the selected patterns. The proposed approach is therefore generic enough to be applied in many domains, and to both text data and networked data.

For future work, I plan to continue designing principled methods to mine networked data with solid theoretical guarantee. In order to make the data mining models more applicable to the real data, I hope to design robust methods by fully considering that the networked data could be noisy, incomplete, dynamic and large-scale. My ultimate goal will be applying the proposed methods to some important applications and products with real impact.

References

[1] S. Agarwal. Ranking on graph data. In *Proceedings of the 23rd international conference on Machine learning*, pages 25–32. ACM, 2006. Cited on Page(s): 9, 44

[2] B. Aleman-Meza, C. Halaschek-Wiener, I. B. Arpinar, and A. P. Sheth. Context-aware semantic association ranking. In *Proceedings of the first International Workshop on Semantic Web and Databases, Co-located with the International Conference on Very Large Data Bases*, pages 33–50, 2003. Cited on Page(s): 9

[3] K. Anyanwu, A. Maduko, and A. Sheth. Semrank: ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on World Wide Web*, pages 117–127. ACM, 2005. Cited on Page(s): 9, 9

[4] K. Anyanwu and A. Sheth. P-queries: enabling querying for semantic associations on the semantic web. In *Proceedings of the 12th international conference on World Wide Web*, pages 690–699. ACM, 2003. Cited on Page(s): 9

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001. Cited on Page(s): 44

[6] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006. Cited on Page(s): 6

[7] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006. Cited on Page(s): 11

[8] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 79–86, 2010. Cited on Page(s): 5, 6, 11

[9] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, pages 19–26, 2001. Cited on Page(s): 11

[10] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998. Cited on Page(s): 20

[11] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. Cited on Page(s): 58

[12] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm. Cited on Page(s): 55

[13] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006. Cited on Page(s): 11

[14] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997. Cited on Page(s): 9, 13, 44, 52

[15] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996. Cited on Page(s): 11, 13

[16] A. Coulet, Y. Garten, M. Dumontier, R. Altman, M. Musen, N. Shah, et al. Integration and publication of heterogeneous text-mined relationships on the semantic web. *Journal of Biomed Semantics*, 2(Suppl 2):S10, 2011. Cited on Page(s): 67

[17] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992. Cited on Page(s): 28

[18] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997. Cited on Page(s): 7

[19] B. Gao, T.-Y. Liu, W. Wei, T. Wang, and H. Li. Semi-supervised ranking on very large graphs with rich metadata. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 96–104. ACM, 2011. Cited on Page(s): 44

[20] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Advances in Neural Information Processing Systems*, pages 585–593, 2009. Cited on Page(s): 37

[21] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996. Cited on Page(s): 15, 18, 49

[22] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 540–547. ACM, 2009. Cited on Page(s): 9

[23] Z. Guan, C. Wang, J. Bu, C. Chen, K. Yang, D. Cai, and X. He. Document recommendation in social tagging services. In *Proceedings of the 19th international conference on World wide web*, pages 391–400. ACM, 2010. Cited on Page(s): 10, 68

[24] A. Guillory and J. A. Bilmes. Label selection on graphs. In *Advances in Neural Information Processing Systems*, pages 691–699, 2009. Cited on Page(s): 1, 6, 6, 6, 6, 11, 12, 20, 20, 22

[25] E. Gunther, D. Stone, R. Gerwien, P. Bento, and M. Heyes. Prediction of clinical drug efficacy by classification of drug-induced genomic expression profiles in vitro. *Science Signalling*, 100(16):9608, 2003. Cited on Page(s): 67

[26] S. Hanneke. An analysis of graph cut size for transductive learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 393–399. ACM, 2006. Cited on Page(s): 11

[27] D. A. Harville. *Matrix Algebra from a Statistician's Perspective*. Springer-Verlag, 1997. Cited on Page(s): 16

[28] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer-Verlag, 2009. Cited on Page(s): 7

[29] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 9–16. ACM, 2004. Cited on Page(s): 9, 44, 55

[30] X. He. Laplacian Regularized D-Optimal Design for Active Learning and Its Application to Image Retrieval. *IEEE Transactions on Image Processing*, 19(1):254–263, 2010. Cited on Page(s): 5

[31] X. He, M. Ji, and H. Bao. A unified active and semi-supervised learning framework for image compression. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72. IEEE, 2009. Cited on Page(s): 2, 5

[32] X. He, M. Ji, C. Zhang, and H. Bao. A variance minimization criterion to feature selection using laplacian regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2026–2038, 2011. Cited on Page(s): 2, 5

[33] S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in neural information processing systems*, pages 892–900, 2010. Cited on Page(s): 5

[34] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002. Cited on Page(s): 67, 72, 75, 80, 86

[35] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM, 2003. Cited on Page(s): 67, 72, 80, 86

[36] M. Ji and J. Han. A variance minimization criterion to active learning on graphs. *Journal of Machine Learning Research - Proceedings Track*, 22:556–564, 2012. Cited on Page(s): 2, 3, 6, 88

[37] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1298–1306. ACM, 2011. Cited on Page(s): 2, 3, 7, 7, 7, 8, 88

[38] M. Ji, B. Lin, X. He, D. Cai, and J. Han. Parallel field ranking. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 723–731. ACM, 2012. Cited on Page(s): 3, 88

[39] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *Proceedings of Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD*, pages 570–586, 2010. Cited on Page(s): 7, 29, 32, 32, 33, 33, 33, 36, 38, 41, 41, 42, 42

[40] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–622, 1999. Cited on Page(s): 7, 26, 44, 44

[41] A. Kuwadekar and J. Neville. Combining semi-supervised learning and relational resampling for active learning in network domains. In *the Budgeted Learning Workshop, International Conference on Machine Learning*, 2010. Cited on Page(s): 5, 6, 6

[42] M. Lades, J. C. Vorbrüggen, J. M. Buhmann, J. Lange, C. von der Malsburg, R. P. Würtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993. Cited on Page(s): 58

[43] J. Lafferty and L. Wasserman. Statistical analysis of semi-supervised regression. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, pages 801–808, 2007. Cited on Page(s): 44

[44] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, 1995. Cited on Page(s): 63

[45] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81:53–67, 2004. Cited on Page(s): 81

[46] N. Lao and W. W. Cohen. Fast query execution for retrieval models based on path-constrained random walks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 881–888. ACM, 2010. Cited on Page(s): 10, 67, 72, 80, 87

[47] B. Lin, C. Zhang, and X. He. Semi-supervised regression via parallel field regularization. In *Advances in Neural Information Processing Systems*, pages 433–441. 2011. Cited on Page(s): 2, 44, 46, 46, 47, 48, 48, 48, 48, 49, 49

[48] D. Liu, X.-S. Hua, L. Yang, M. Wang, and H.-J. Zhang. Tag ranking. In *Proceedings of the 18th international conference on World wide web*, pages 351–360. ACM, 2009. Cited on Page(s): 9

[49] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, pages 585–592. ACM, 2006. Cited on Page(s): 32

[50] J. Long, J. Yin, W. Zhao, and E. Zhu. Graph-based active learning based on label propagation. In *Modeling Decisions for Artificial Intelligence*, pages 179–190. Springer, 2008. Cited on Page(s): 5, 5

[51] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning*, 2003. Cited on Page(s): 6

[52] S. A. Macskassy. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 597–606. ACM, 2009. Cited on Page(s): 5, 5, 6, 21

[53] S. A. Macskassy and F. Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining at the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 64–76, 2003. Cited on Page(s): 6, 36

[54] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007. Cited on Page(s): 26, 36, 36, 37, 37

[55] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Cited on Page(s): 56, 59, 83

[56] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007. Cited on Page(s): 6

[57] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 2, pages 849–856. MIT; 1998, 2002. Cited on Page(s): 20

[58] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: bringing order to web objects. In *Proceedings of the 14th international conference on World Wide Web*, pages 567–574. ACM, 2005. Cited on Page(s): 7

[59] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996. Cited on Page(s): 58

[60] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998. Cited on Page(s): 7, 26, 44, 44

[61] K. Pelckmans, J. Shawe-Taylor, J. A. K. Suykens, and B. D. Moor. Margin based transductive graph cuts using linear programming. *Journal of Machine Learning Research - Proceedings Track*, 2:363–370, 2007. Cited on Page(s): 11

[62] P. Petersen. *Riemannian Geometry*. Springer, 1998. Cited on Page(s): 45, 45, 46, 47, 47, 47

[63] C. Ramakrishnan, P. Mendes, S. Wang, and A. Sheth. Unsupervised discovery of compound entities for relationship extraction. *Knowledge Engineering: Practice and Patterns*, pages 146–155, 2008. Cited on Page(s): 67

[64] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. Cited on Page(s): 44

[65] D. Searls. Data integration: challenges for drug discovery. *Nature Reviews Drug Discovery*, 4(1):45–58, 2005. Cited on Page(s): 67

[66] P. Sen and L. Getoor. Link-based classification. Technical Report CS-TR-4858, University of Maryland, February 2007. Cited on Page(s): 6, 26, 36

[67] S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web*, pages 671–680. ACM, 2009. Cited on Page(s): 10, 68

[68] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2010. Cited on Page(s): 5, 11

[69] A. P. Sheth, B. Aleman-Meza, I. B. Arpinar, C. Bertram, Y. S. Warke, C. Ramakrishnan, C. Halaschek, K. Anyanwu, D. Avant, F. S. Arpinar, and K. Kochut. Semantic association identification and knowledge discovery for national security applications. *Journal of Database Management (JDM)*, 16(1):33–53, 2005. Cited on Page(s): 9

[70] C. Shi, X. Kong, P. S. Yu, S. Xie, and B. Wu. Relevance search in heterogeneous networks. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 180–191. ACM, 2012. Cited on Page(s): 10, 67, 68, 72, 80, 87

[71] L. Shi, Y. Zhao, and J. Tang. Combining link and content for collective active learning. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1829–1832. ACM, 2010. Cited on Page(s): 6

[72] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618, 2003. Cited on Page(s): 58

[73] L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–676. ACM, 2008. Cited on Page(s): 6

[74] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of 2011 International Conference on Very Large Data Bases*, 4(11):992–1003, 2011. Cited on Page(s): 4, 10, 67, 68, 68, 72, 75, 81

[75] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 797–806. ACM, 2009. Cited on Page(s): 7, 29, 31, 37, 39

[76] D. Tao, X. Li, X. Wu, and S. J. Maybank. Geometric mean for subspace selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):260–274, 2009. Cited on Page(s): 44

[77] D. Tao, X. Tang, X. Li, and Y. Rui. Direct kernel biased discriminant analysis: A new content-based image retrieval relevance feedback algorithm. *IEEE Transactions on Multimedia*, 8(4):716–727, 2006. Cited on Page(s): 44

[78] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. Cited on Page(s): 44

[79] S. Tong and E. Y. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the 9th ACM International Conference on Multimedia*, pages 107–118, 2001. Cited on Page(s): 55

[80] X. Wan, J. Yang, and J. Xiao. Manifold-Ranking Based Topic-Focused Multi-Document Summarization. In *International Joint Conference on Artificial Intelligence*, pages 2903–2908, 2007. Cited on Page(s): 44

[81] B. Xu, J. Bu, C. Chen, D. Cai, X. He, W. Liu, and J. Luo. Efficient manifold ranking for image retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 525–534, 2011. Cited on Page(s): 55, 56

[82] S. Yan, W. S. Spangler, and Y. Chen. Cross media entity extraction and linkage for chemical documents. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011. Cited on Page(s): 73

[83] D. Yin, Z. Xue, L. Hong, and B. Davison. A probabilistic model for personalized tag prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–968. ACM, 2010. Cited on Page(s): 10, 68

[84] X. Yuan, X.-S. Hua, M. Wang, and X.-Q. Wu. Manifold-ranking based video concept detection on large database and feature pool. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 623–626. ACM, 2006. Cited on Page(s): 44

[85] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks*, 31(11-16):1361–1374, 1999. Cited on Page(s): 28

[86] Z.-J. Zha, M. Wang, Y.-T. Zheng, Y. Yang, R. Hong, and T.-S. Chua. Interactive video indexing with statistical active learning. *IEEE Transactions on Multimedia*, 14(1):17–27, 2012. Cited on Page(s): 5

[87] Y. Zhang and Z. hua Zhou. Non-Metric Label Propagation. In *International Joint Conference on Artificial Intelligence*, pages 1357–1362, 2009. Cited on Page(s): 6

[88] W. Zhao, J. Long, E. Zhu, and Y. Liu. A scalable algorithm for graph-based active learning. In *Proceedings of 2008 International Workshop on Frontiers in Algorithmics (FAW'08), (Springer Lecture Notes in Computer Science 5059), Changsha, China, June 2008*, pages 311–322, 2008. Cited on Page(s): 5, 5

[89] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2003. Cited on Page(s): 6, 32, 36, 41

[90] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. *Advances in Neural Information Processing Systems*, 16:169–176, 2003. Cited on Page(s): 2, 8, 8, 8, 8, 9, 33, 44, 44, 44, 48, 55, 55, 63

[91] X. Zhou, M. Belkin, and N. Srebro. An iterated graph laplacian approach for ranking on manifolds. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 877–885. ACM, 2011. Cited on Page(s): 9, 9, 44, 44

[92] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *International Conference on Machine Learning*, pages 912–919, 2003. Cited on Page(s): 5, 6, 11, 11, 12, 13, 13, 20

[93] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *the workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, International Conference on Machine Learning*, pages 58–65, 2003. Cited on Page(s): 5, 5, 6, 6, 11, 12, 13, 13